Proyecto fin de carrera

Herramienta de ayuda al análisis forense en vivo de sistemas Linux

Ingeniería Informática Superior

Universidad Carlos III de Madrid

Autor: José María López Otero

Tutor: Juan Manuel Canelada Oset

Fecha: 23 de Septiembre de 2011

Agradecimientos

Siempre me he considerado pésimo para discursos sentimentales y he dejado esos minutos de protagonismo a otras personas (véase mi gran amigo y compañero de carrera Diego), pero creo que llegados a este punto sólo yo puedo tomar estas riendas.

A los primeros a los que me quiero dirigir son a mis padres, José María y Rosa, ya que gracias a su esfuerzo diario he recibido todo lo que he necesitado en la vida: cariño y educación. Ellos, junto a mis hermanos, Martina y Víctor, han sido los pilares de mi vida. Gracias de corazón.

Quiero hacer una especial mención a mis sobrinos Álvaro y Hugo, cuyas vidas están comenzando. Espero que tengan la suerte que yo tuve con mis padres y puedan estudiar lo que les guste para poder trabajar en lo que quieran. A ellos les dejo esta frase de Confucio: "*Escoge un trabajo que te guste y nunca tendrás que trabajar ni un solo día de tu vida*". Confío en que, con el tiempo, la entenderán.

Aunque parezca mentira, también quiero dirigir parte de mis agradecimientos a la Universidad Carlos III de Madrid. Sin ella no hubiera podido conocer a lo largo de todos los años a ese magnífico grupo de compañeros que hicimos piña y nos apodamos Cebollitas. Tantas horas de prácticas, encerrados en las aulas de informática del edificio Sabatini, Betancourt o Torres Quevedo, unidos a las numerosas fiestas, viajes y brindis, sirven para estrechar lazos y poder decir con una amplia sonrisa que son mis amigos. Siempre os llevaré dentro.

A mi queridísimo tutor, Juan Manuel. De él he aprendido muchas cosas, pero si pudiera resumirlas en una sola frase sería la siguiente: la perfección no existe, pero sí el camino a ella, por lo que siempre podremos esforzarnos por ser mejores.

A todo el departamento del Servicio de Informática de Leganés, en el que llevo tres años y mi viaje con ellos toca su fin, pero en especial a mi jefecillo, Nacho. Siempre ha sido para mí como una luz en un sendero oscuro. Agradezco todos los consejos que me ha dado y espero que se los pueda transmitir con la misma facilidad a su hijo, Lucas.

Espero que me disculpen el sin fin de personas que no aparecen en estas líneas y han contribuido de alguna manera en mi vida y en el desarrollo de este proyecto. A todos ellos, anónimos míos, perdón y gracias.

Mi último agradecimiento va dirigido a la persona más importante de mi vida, a una chica que me hechizó desde el primer instante en el que mi corazón tuvo conocimiento de su existencia. Ella es tan especial y hermosa como su nombre, Arlanza. Si hay algo que pueda decir por todo lo que ha contribuido a mi vida es, sencillamente, gracias. Por los momentos que hemos vivido. Por los que nos quedan por vivir. Por los buenos que me hacen sonreír. Por los malos que me hacen reflexionar. Por enseñarme cada día una lección nueva que me hace mejor persona. Por permitirme poner mi vida junto a la suya. Por ser capaz de sacarme una sonrisa en los peores momentos. Por ti. Por mí. Por nosotros.

Resumen

Cada día se descubren nuevas vulnerabilidades en los sistemas operativos y, de manera intrínseca, formas de explotar esas vulnerabilidades.

El objetivo principal de este proyecto consiste en el análisis, diseño e implementación de una herramienta desarrollada en Java que sirva de ayuda al análisis forense en vivo de sistemas Linux. Para ello presentará, de forma gráfica y en tiempo real, información de la máquina en la que se ejecuta, referente a procesos, conexiones de red, módulos cargados y conexiones SSH establecidas.

Como resultado se ha obtenido una aplicación de análisis con las siguientes características:

- **No modificable:** está pensada para ser ejecutada desde dispositivos de sólo lectura y así evitar posibles infecciones de malware.
- **Autocontenida:** no llama a ejecutables del sistema operativo que pueden estar comprometidos, sino que trabaja con la información del kernel que procesa a través de /proc.
- **Portable:** gracias a los dos puntos anteriores conseguimos una herramienta funcional sin necesidad de instalación.

Palabras clave: seguridad, análisis, forense, Java, aplicación, portable.

Abstract

New vulnerabilities and different ways to exploit them on operative systems are discovered every day.

The main targets of this project are the analysis, design and implementation of a Java developed tool, which helps us with the forensic analysis of Linux live systems.

To accomplish this task, the application will display, graphically and in real time, information about the computer in which the tool is being executed. These data will inform about processes, network connections, kernel loaded modules and established SSH connections.

The obtained result is an application with the following features:

- **Unmodificable:** the tool must be executed from read-only devices in order to avoid malware attacks.
- **Self-contained:** instead of calling system executables that could be already corrupt it works with the kernel information that is processed through /proc.
- **Portable:** thanks to the previous features, we get a functional tool that doesn't need to be installed.

Keywords: security, analysis, forensic, Java, application, portable.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

El presente documento describe las fases de desarrollo de la herramienta desarrollada para este proyecto, una aplicación desarrollada en Java para el análisis forense de máquinas Linux. Presenta de forma gráfica información en vivo de la máquina en la que se ejecuta referente a procesos, conexiones de red, módulos cargados y conexiones SSH establecidas.

1.1. Motivación

El mundo de la informática cada vez es menos seguro, y como decía don Francisco de Quevedo: "No vive el que no vive seguro". Cada día se descubren nuevas vulnerabilidades en los sistemas operativos y, de manera intrínseca, formas de explotar esas vulnerabilidades.

Según un informe de RedIRIS [1], los incidentes de seguridad ocurridos en 2010 que afectaron a las redes académicas y de investigación españolas ascienden a 5337.

En la siguiente gráfica se muestra la evolución de los incidentes atendidos por IRIS-CERT[1] desde el año 1999.

[1] Servicio de seguridad que da soporte a las instituciones afiliadas a RedIRIS. Más información en *http://www.rediris.es/cert/*.

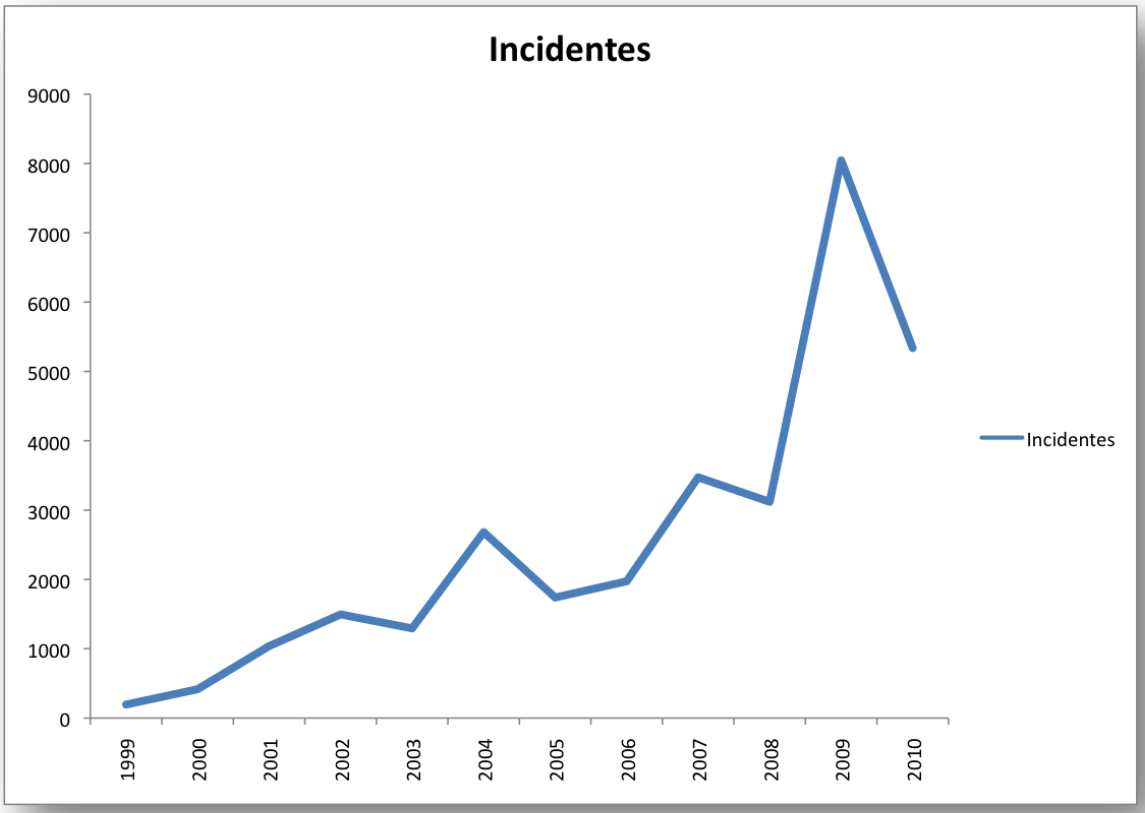


Ilustración 1: Evolución de incidentes de seguridad

Se puede apreciar un claro aumento a medida que avanzan los años por lo que es necesario tomar medidas para frenar esta oleada de ataques. Una solución factible es dotar a los administradores de sistemas de medios para obtener información de las máquinas comprometidas con el fin de detectarlos y evitar que se produzcan.

En la siguiente gráfica se puede ver el número de incidentes que se recogieron por mes a lo largo de 2010.

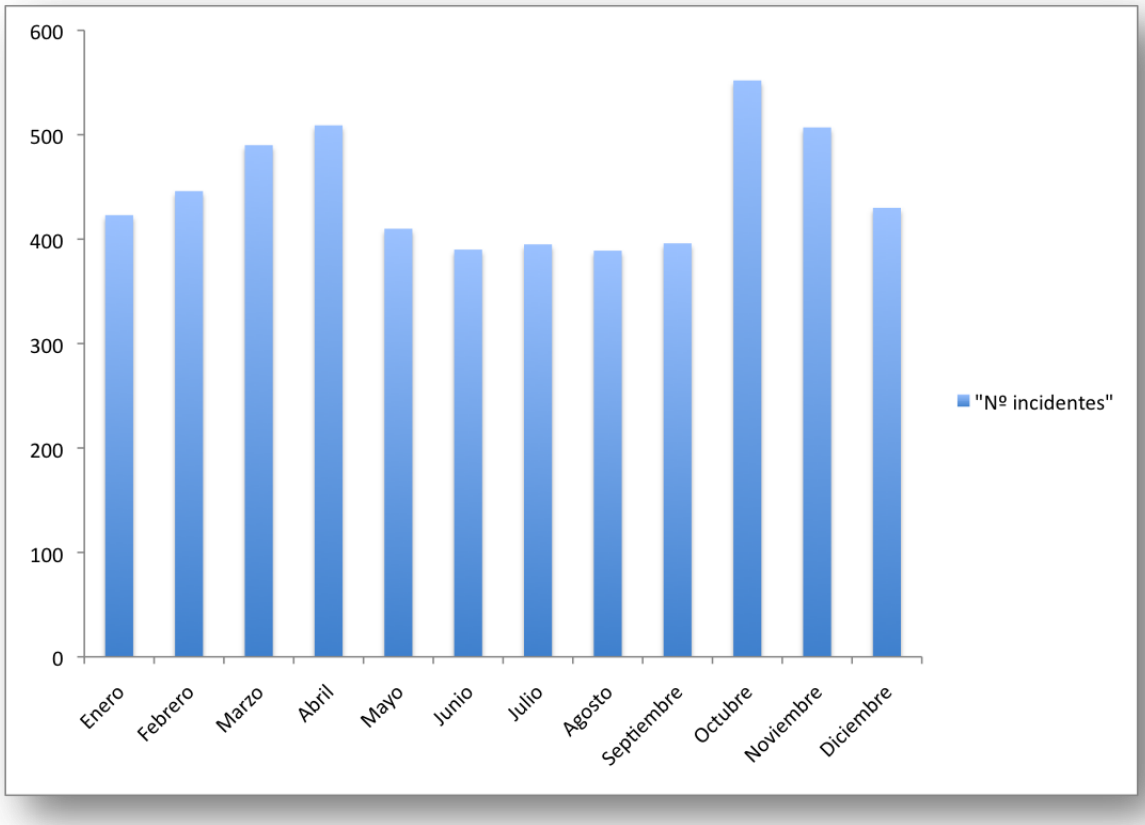


Ilustración 2: Incidentes por mes en 2010

Según el estudio de RedIRIS se recibieron una media de 444,75 incidentes por mes en 2010. Lo que totaliza una media de 14,62 incidentes por día.

En la siguiente gráfica se muestra la distribución de los incidentes según su taxonomía recibidos en 2010.

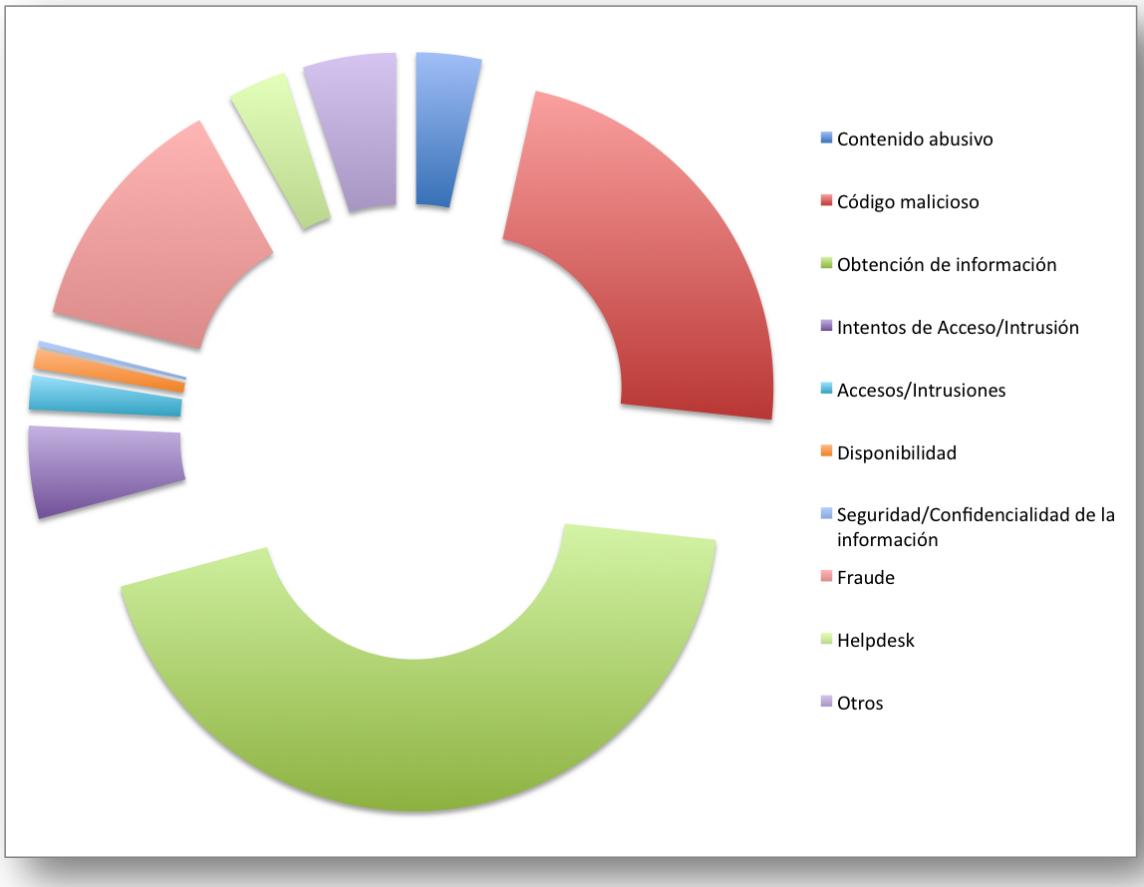


Ilustración 3: Incidentes según su taxonomía en 2010

Según su taxonomía tuvieron como objetivo principal obtener información mediante la instalación de malware[2], fraude e intentos de acceso no autorizado.

Las principales herramientas para combatir malware son reactivas, depende de una base de datos que necesita una constante actualización, y son fáciles de burlar con cierto grado de conocimiento. Es necesario un análisis a más bajo nivel (procesos, conexiones, módulos) que consiga abstraernos de su estructura física (qué es) y centrarnos más en su comportamiento (qué hace).

Esta filosofía frente a programas maliciosos sufre cierta ruptura cuando se trata de un rootkit[3]. Los hay de dos tipos: de kernel y de usuario. La finalidad de ambos es la misma, esconder información sobre el intruso. Los de kernel son más difíciles de detectar porque

[2] Software malicioso que tiene como objetivo infiltrarse o dañar una máquina sin el consentimiento de su propietario.

[3] Un rootkit es una herramienta que tiene como finalidad esconderse a sí misma y a otros programas, procesos, archivos, directorios, claves de registro, conexiones de red, etc., con el fin de comandar acciones o extraer información sensible remotamente.

modifican una parte del código del núcleo, comúnmente, mediante un módulo en Linux o un controlador en Windows. Los de tipo usuario pueden reemplazar los archivos ejecutables que leen información del núcleo, por lo tanto accediendo directamente a las estructuras del kernel se podrían detectar.

Gracias disponer del código fuente se puede analizar cómo el kernel[4] de Linux gestiona los recursos de la máquina. Como valor añadido, la comunidad de desarrolladores proporciona documentación actualizada accesible vía Internet. Por lo tanto, resulta menos tedioso el desarrollo de aplicaciones de sistema para un sistema operativo de código abierto que para uno de caja negra como Windows.

En Linux existe una infinidad de programas en modo texto para analizar el sistema (*ps, lsmod, lsof, netstat, grep* para buscar en ficheros de log…). Todos muestran gran cantidad de información y son muy potentes si el usuario los conoce en profundidad. El problema es que no siempre el usuario sabe desenvolverse bien en este entorno que puede resultar rutinario para un analista de seguridad, por lo que surge la necesidad de una herramienta gráfica, amigable y sencilla que pueda reunir toda la información que necesita el usuario para realizar un análisis previo del sistema en el que se encuentra.

En la Universidad Carlos III de Madrid (de ahora en adelante UC3M), al igual que en otras universidades y empresas, se ha impulsado el software libre por su transparencia y para abaratar costes. Los principales servidores de correo, web, directorio, entornos de desarrollo, corren bajo un Linux, por lo que se necesita una herramienta de análisis, gráfica y sencilla, para este tipo de máquinas.

[4] En informática, el kernel es el núcleo del sistema operativo.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación gráfica que muestre información en vivo de la máquina en la que se ejecuta y que pueda ser utilizada en su análisis forense. Información referente a procesos, conexiones de red, módulos[5] cargados por el sistema operativo y conexiones SSH establecidas. Se quiere utilizar en máquinas presuntamente asaltadas lo cual implica una serie de medidas de seguridad a seguir.

La aplicación debe de cumplir los siguientes requisitos fundamentales:

- Ejecutable en sistemas Linux: la aplicación se quiere ejecutar en máquinas Linux, ya que en la actualidad los principales comandos, como se verá, trabajan en modo texto para obtener información. Dichos comandos requieren especialización por parte del usuario.
- Ejecutable desde dispositivos de sólo lectura: como está aplicación está pensada para probar en máquinas asaltadas, un requisito fundamental es que sea ejecutada desde dispositivos de sólo lectura como CD-ROMs, DVD-ROMs, pendrives con modo sólo lectura, etc. De esta forma se garantiza que el dispositivo de sólo lectura, y por tanto la aplicación, permanecen íntegros frente al malware que pudiera contener la máquina.
- Autocontenida y portable: para que se cumpla el punto anterior la aplicación debe ser necesariamente portable, es decir, que se pueda ejecutar sin necesidad de una instalación previa. Además debe ser autocontenida en la medida de lo posible, es decir, que debe valerse de sí misma para presentar la información al usuario sin tener que utilizar programas externos. Esta medida de precaución se debe a que los programas del sistema podrían estar comprometidos, mediante rootkits de usuario, y no mostrar información veraz.

[5] Un módulo es un controlador de un dispositivo (driver en Windows) o un servicio que puede cargarse o descargarse cuando el usuario o algún dispositivo lo solicita.

1.3. Estructura del documento

Este apartado describe la composición de los capítulos que conforman la totalidad del presente documento con el objetivo de facilitar la lectura del mismo.

1. **Introducción:** supone una primera toma de contacto con el documento. Sitúa en contexto al lector hablando de la motivación y los objetivos del proyecto.
2. **Estado del arte:** describe la situación actual en lo que respecta a las herramientas de análisis de seguridad en Linux. Se habla de la aplicación a desarrollar y de los aportes que hace al campo de la seguridad.
3. **Gestión del proyecto:** describe una planificación inicial de las tareas a desempeñar y un presupuesto ligado para el desarrollo del proyecto.
4. **Análisis:** comprende la fase de análisis del ciclo de desarrollo de software. Comprende las necesidades a descubrir y su transformación en requisitos.
5. **Diseño:** abarca la fase de diseño del ciclo de desarrollo de software.
6. **Implementación:** este capítulo comprende la fase de implementación del ciclo de desarrollo de software.
7. **Pruebas:** en este apartado se recogen las pruebas unitarias, de integración y de sistema realizadas para este proyecto.
8. **Conclusiones y líneas futuras:** describe las conclusiones alcanzadas mediante la elaboración del proyecto y una serie de sugerencias a tomar como líneas futuras en el desarrollo del análisis de máquinas asaltadas.
9. **Anexos:** recoge información adicional de interés para el lector.
10. **Bibliografía:** enumera la bibliografía consultada para la elaboración del proyecto.
11. **Referencias:** por último, en este punto se recogen las referencias utilizadas en el desarrollo del proyecto y en la redacción del presente documento.

2. Estado del arte

El estado del arte tiene por objetivo presentar el entorno tecnológico actual sobre herramientas de monitorización en sistemas Unix[6] y en sistemas Windows. Se comentarán las herramientas que integran cada sistema operativo de forma nativa y otras que no se incluyen por defecto pero que son de fácil acceso. Se comentará la herramienta de integración What's Running[7] desarrollada para sistemas Windows, en la que se basa en gran parte este proyecto. Por último, se concluirá el capítulo con el vacío y las necesidades que existen en sistemas Linux y cómo el desarrollo de este proyecto pretende solventarlas.

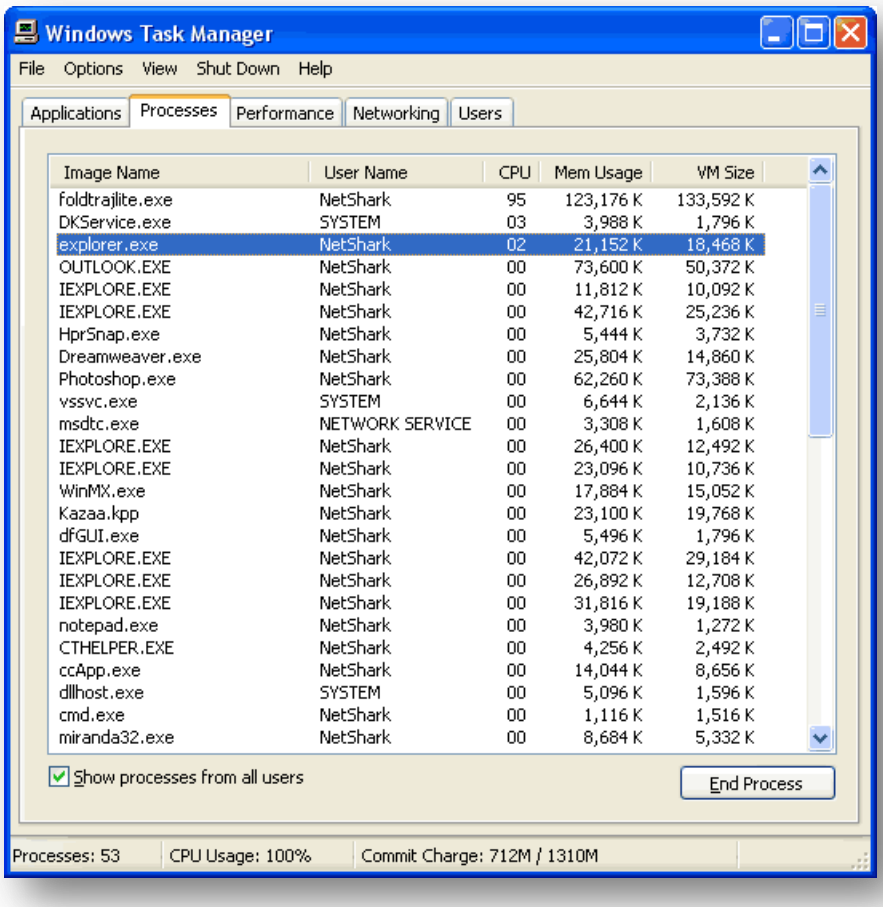


Ilustración 4: Administrador de tareas de Windows XP

[6] Cuando se hable de sistemas Unix se refiere al marco global de sistemas operativos basados en ese sistema, de los cuales Linux es uno de ellos.

[7] Software de monitorización para Windows. Más información en su página oficial http://www.whatsrunning.net/.

2.1. Herramientas de monitorización

a) Herramientas de monitorización en Linux

Los sistemas Unix integran de forma nativa numerosas herramientas que aportan información del sistema. Son herramientas de línea de comandos, no poseen modo gráfico y, aunque son muy potentes, pueden resultar un poco tediosas a la hora de usar.

Por ejemplo supongamos un escenario en el que necesitemos usar varias de estas herramientas de interfaz textual, lo que supondría:

- Recordar las opciones de los comandos deseados.
- En caso de no recordarlas recurrir a la documentación.
- Filtrar la salida de los programas de manera artesana.

A continuación se describen algunas de las herramientas que traen nativas los sistemas Unix[8]:

[8] Puede acceder a la documentación de cada comando accediendo a las páginas del manual desde una terminal de Unix escribiendo "man <comando>". También las puede encontrar en la web *UnixHelp for Users*

- ps

Process status. Muestra un informe que contiene una instantánea de los procesos que se están ejecutando actualmente en el sistema. La siguiente ilustración muestra un fragmento de la salida generada por el programa ps con las opciones “aux”. Con esas opciones se logran mostrar todos los procesos del sistema con la sintaxis BSD.

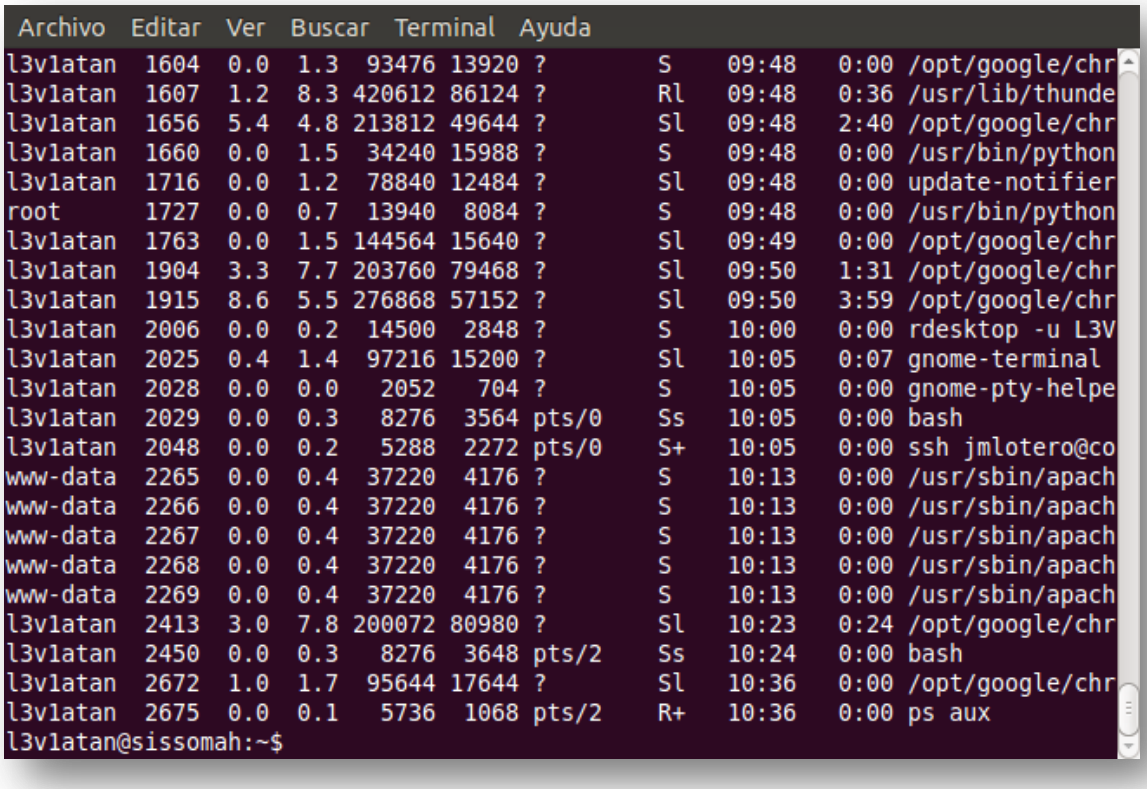


Ilustración 5: Captura del programa ps

- netstat

Network statistics. Muestra un listado de las conexiones activas de la máquina en la que se ejecuta. Posee muchas opciones interesantes entre las que se puede destacar mostrar direcciones numéricas (sin resolver) o mostrar sockets de escucha.

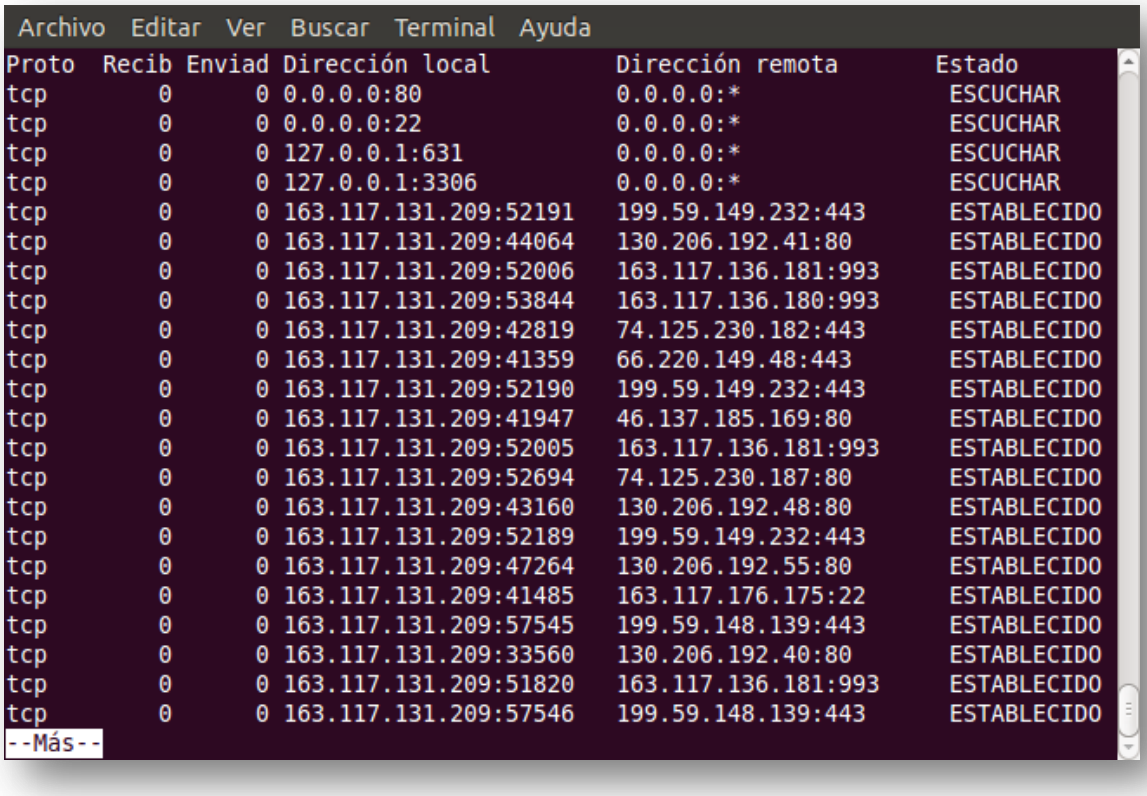


Ilustración 6: Captura del programa netstat

- lsmod

List loaded modules. Muestra información sobre todos los módulos cargados en el kernel. Muestra el nombre del módulo, el tamaño en memoria y el número de módulos que están usando ese módulo seguido de sus nombres.

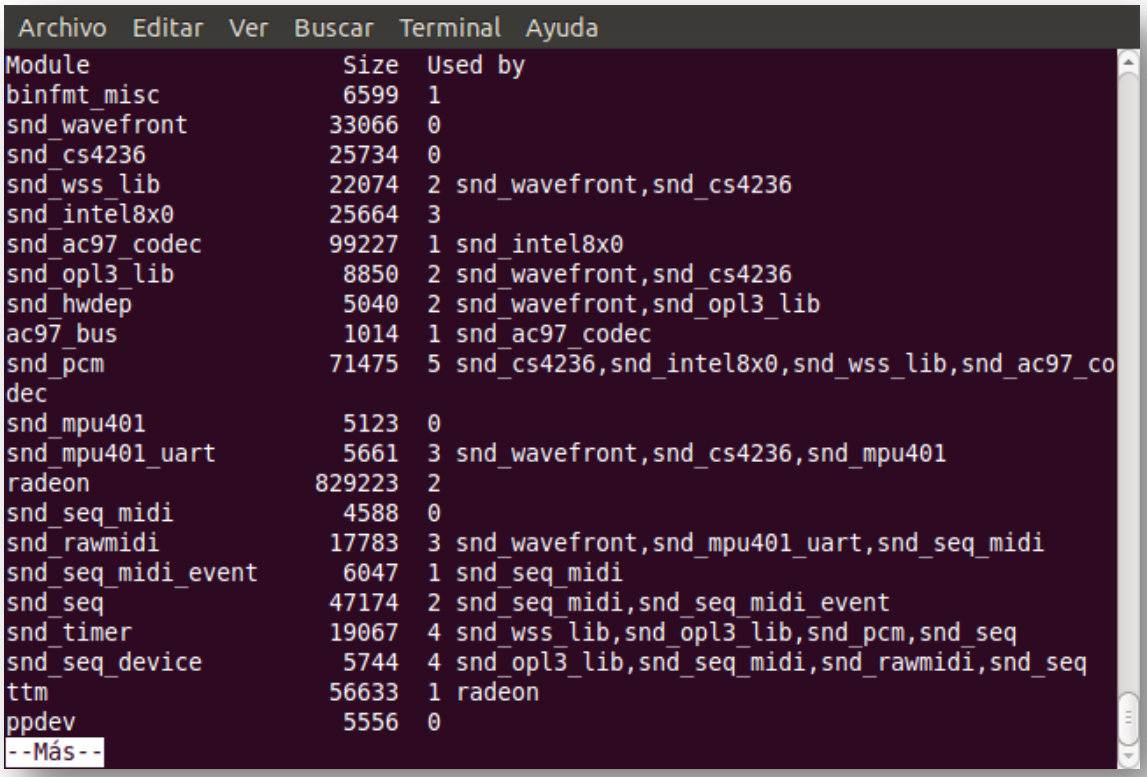


Ilustración 7: Captura del programa lsmod

- lsof

List open files. Muestra todos los archivos de disco que mantiene abiertos un determinado proceso, incluyendo los sockets de red abiertos y tuberías. Este programa posee muchas opciones interesantes, por ejemplo la opción “i”. Con ella se pueden filtrar todos los programas que están usando alguna conexión de red. También permite hacer filtrados para un especificado protocolo o puerto (por ejemplo i:80).

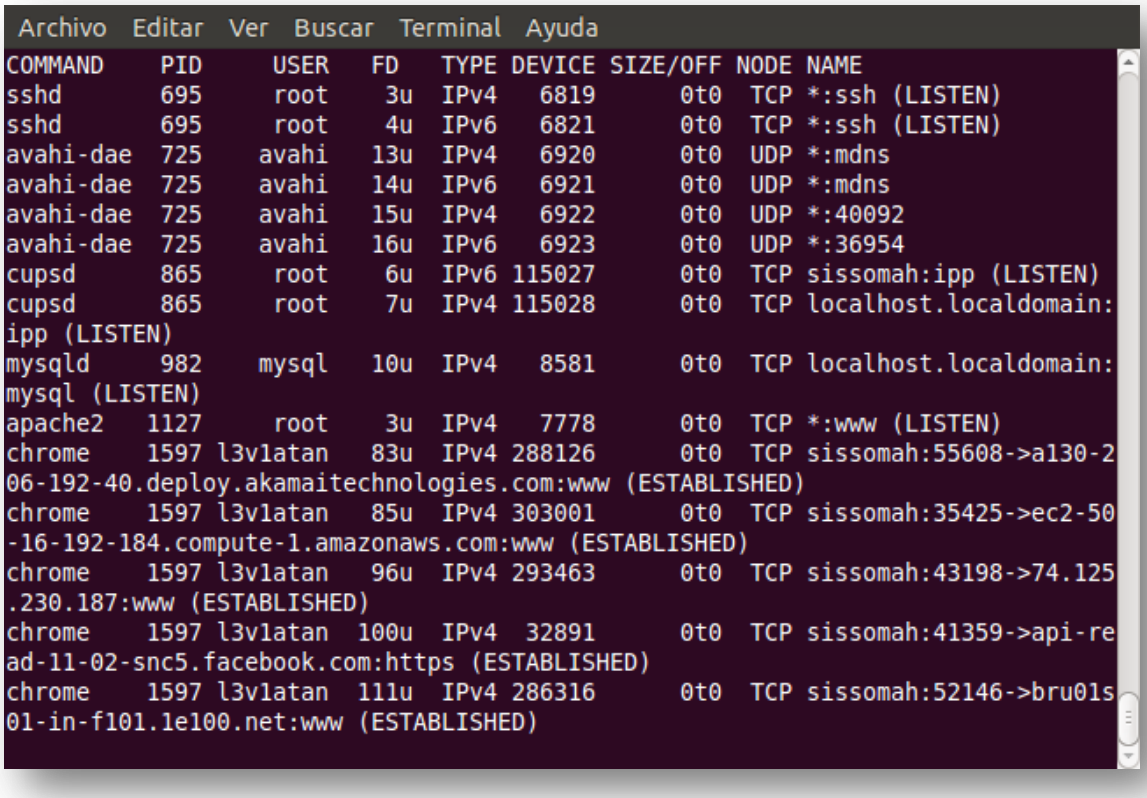


Ilustración 8: Captura del programa lsof

- Ficheros de log

Desde el enfoque de la monitorización de sistemas cabe hacer una mención especial a los ficheros de log. En Unix se suelen almacenar bajo la ruta /var/log. En esa ruta podemos encontrar distintos ficheros en los que se almacenan los mensajes generados por programas, aplicaciones y demonios. Cada línea de log suele contener el nombre del proceso que lo generó, un mensaje (por ejemplo “login failed”) y una fecha-hora.

En Debian/Ubuntu, un fichero de log importante es /var/log/auth.log. Contiene todos los mensajes de log relacionados con las autorizaciones de programas como su, passwd, sshd.

En la siguiente ilustración se puede observar un fragmento de un fichero /var/log/auth.log.

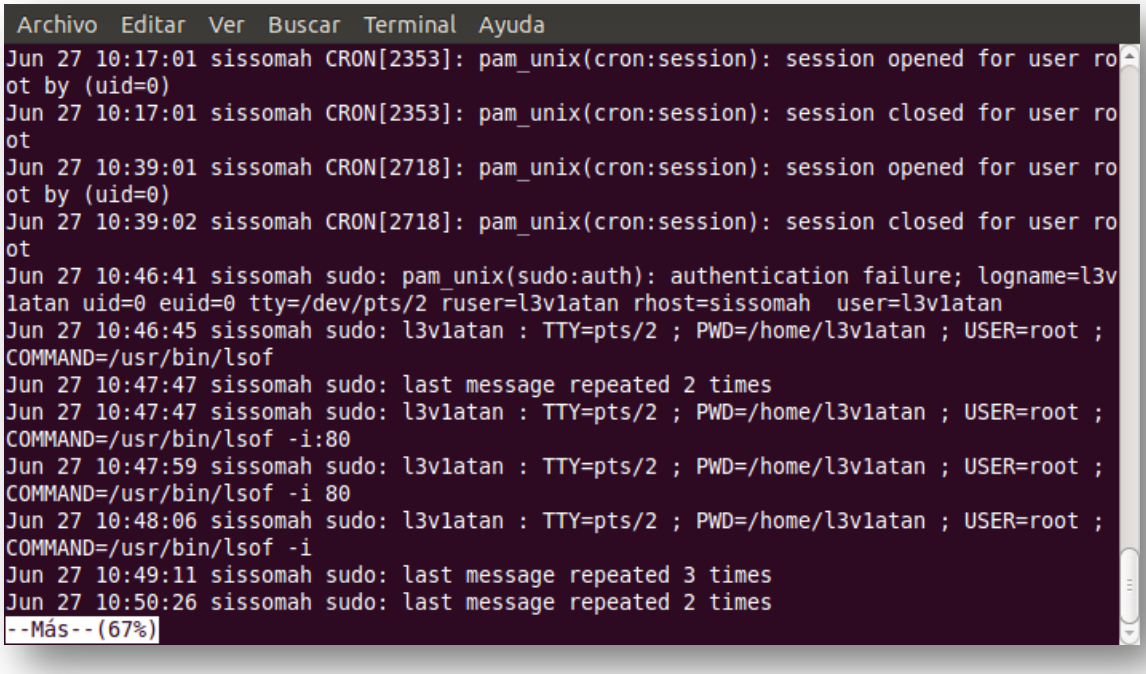


Ilustración 9: Fragmento de un fichero de log

b) Herramientas de monitorización en Windows

En Windows se poseen menos herramientas de líneas de comandos nativas que en Linux. De las aplicaciones citadas en el apartado anterior algunas tienen su análogo en Windows:

- netstat

Network statistics. Muestra un listado de las conexiones activas de la máquina en la que se ejecuta. Es bastante parecido al comando de Linux que se explicó en el anterior apartado. Con la opción "?" muestra la ayuda del programa.

- tasklist

Task list. Es el análogo al programa ps de Unix. En Windows el concepto de task (tarea) es similar al de process (proceso) en Linux. Con la opción "/?" muestra la ayuda del programa.

De forma no nativa, desde 2006 Windows proporciona un conjunto de aplicaciones de sistema avanzadas llamado Sysinternals [2].

- Sysinternals

En 1996, Mark Russinovich y Bryce Cogswell crearon el sitio web de Sysinternals donde alojarían sus herramientas para diagnóstico de sistemas avanzados. Fue en 2006 cuando Windows adquirió Sysinternals.

En la web de Sysinternals podemos encontrar que las utilidades están divididas en seis apartados:

1. Utilidades de disco y archivos:
   Utilidades para ver y supervisar el acceso a los discos y archivos, así como el uso que se hace de ellos.
2. Red:
   Herramientas de red, desde monitores de conexión hasta analizadores de seguridad de recursos.
3. Procesos y subprocesos:
   Utilidades para consultar de forma subyacente los procesos en ejecución y los recursos que consumen.
4. Utilidades de seguridad:

Utilidades de administración y configuración del sistema de seguridad, con programas de rootkit y de captura de spyware.

5. Información del sistema:

   Utilidades para consultar el uso y la configuración de los recursos del sistema.

6. Varios:

   Una colección de utilidades variadas que incluye un protector de pantalla, ayuda de presentación y una herramienta de depuración.

2.2. Herramientas de integración en sistemas Windows: What's Running

Hasta el momento se han mencionado herramientas que analizan una o pocas funcionalidades del sistema. What's Running [3] es un software para Windows que integra varias de las funcionalidades citadas anteriormente.

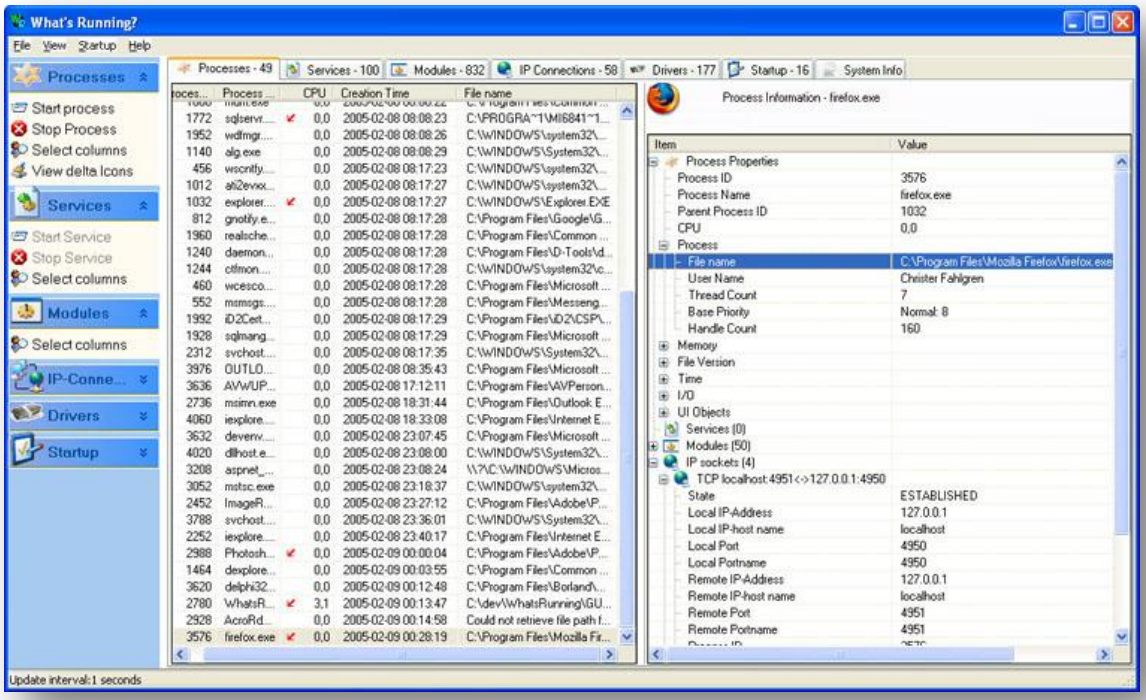


Ilustración 10: What's Running

Esta herramienta muestra información fresca del sistema agrupándola en las siguientes siete pestañas:

a) Procesos:

   Muestra un listado de procesos en ejecución. Para cada uno se puede obtener un detalle.

b) Conexiones IP:

   Muestra un listado de las conexiones IP activas en el sistema. Muestra las conexiones remotas de cada programa y aplicaciones que están escuchando nuevas conexiones.

c) Servicios:

   Muestra un listado de servicios en ejecución y parados.

d) Módulos:

Muestra información sobre librerías y ejecutables que se están usando en el sistema. Para cada módulo se muestran todos los procesos que han cargado ese módulo.

e) Drivers:

Muestra información sobre todos los drivers del sistema. En los drivers en ejecución se puede encontrar el proveedor del driver.

f) Startup:

Muestra y permite administrar los programas que se ejecutan al arranque del sistema operativo.

g) Información del sistema:

Muestra información del sistema sobre el hardware y el sistema operativo.

El principal problema encontrado en What's Running es que solamente es compatible con Windows (para versiones 2000 y superiores).

2.3. Un vacío a cubrir en Linux

Una vez vistas las herramientas con las que se cuenta en Linux y las soluciones existentes en Windows se puede observar un vacío en los sistemas Linux, la falta de una solución gráfica que aúne las funcionalidades de las herramientas textuales.

Ubuntu integra de forma nativa los programas Log File Viewer [9] y System Monitor[10]. En definitiva son herramientas muy sencillas que realmente no cumplen con los requisitos para realizar un análisis exhaustivo de seguridad. Log File Viewer se limita a mostrar el contenido de los ficheros de log, siendo realizar cualquier tipo de ordenación y mezcla los log de distintos programas. System Monitor no muestra las conexiones de red y presenta los procesos con poco detalle.

A continuación se muestran unas ilustraciones de dichos programas:

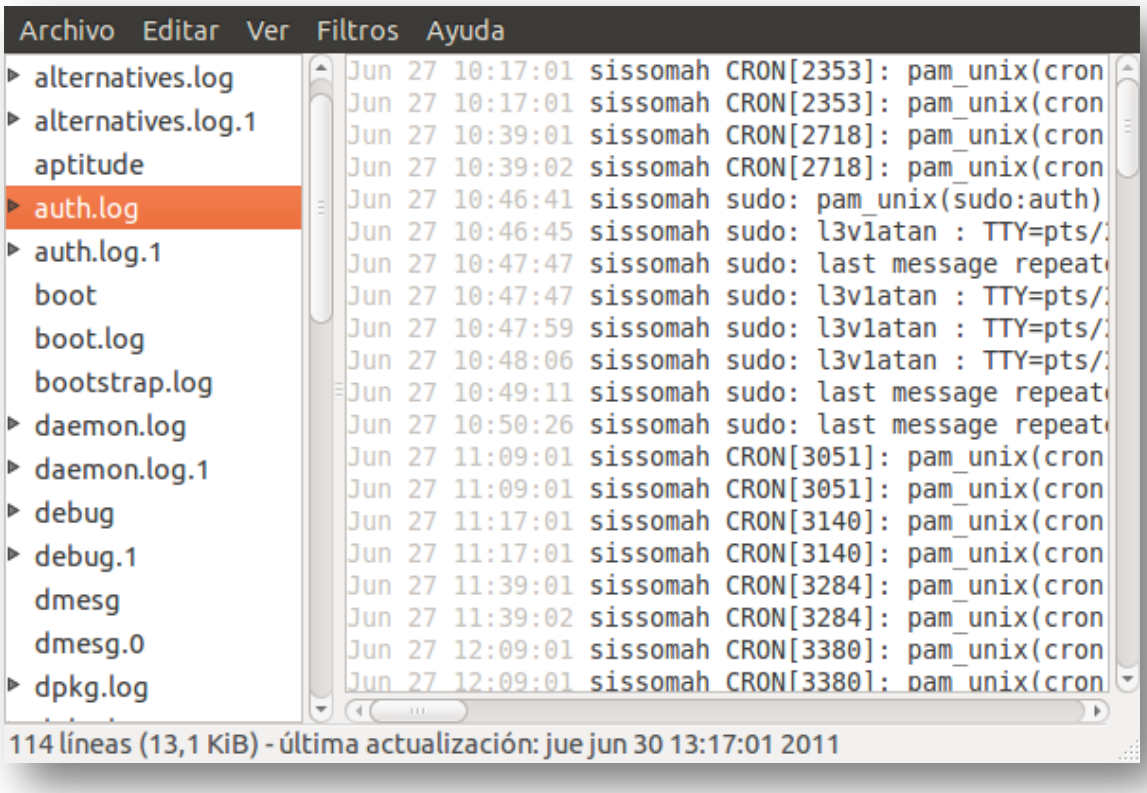


Ilustración 11: Log File Viewer

[9] Visor de ficheros de log. En Ubuntu se encuentra en el menú Sistema > Administración > Visor de archivos de sucesos.

[10] Monitor del sistema. En Ubuntu se encuetra en el menú Sistema > Administración > Monitor del sistema

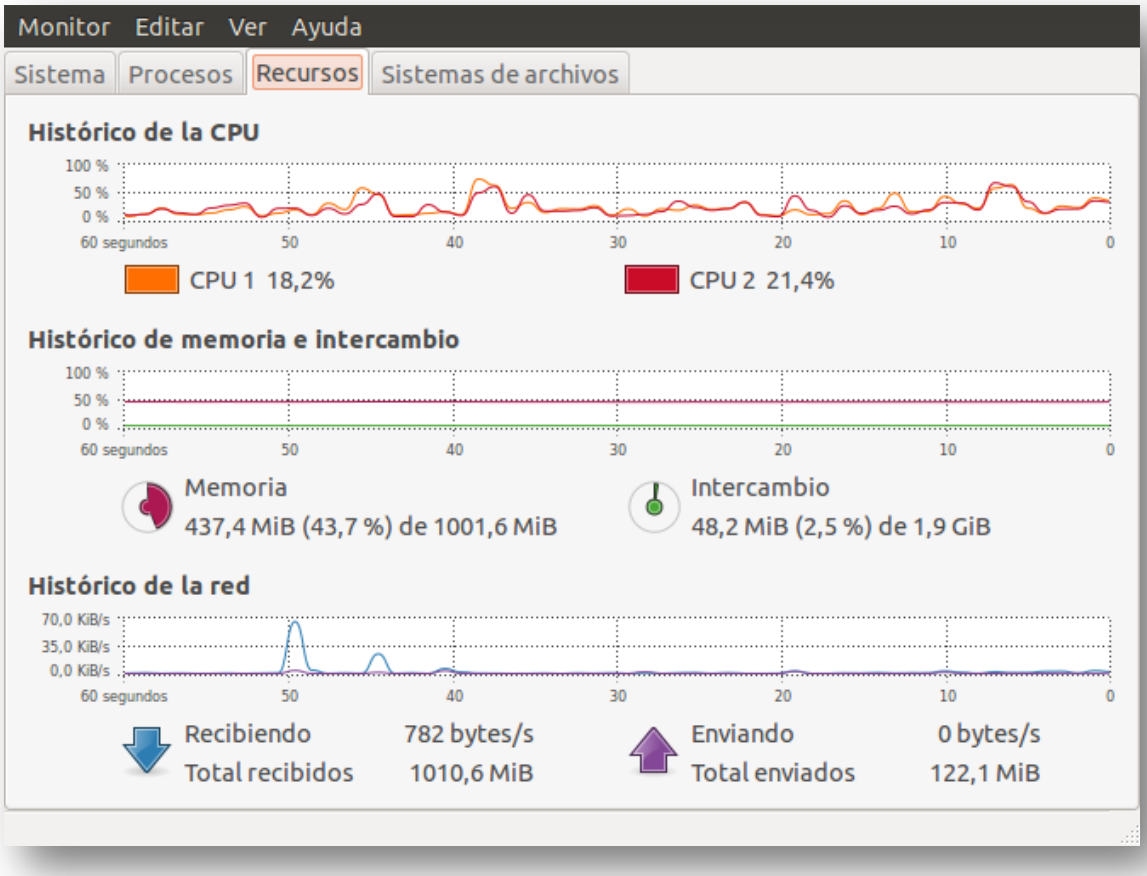


Ilustración 12: System Monitor

Estas aplicaciones no son lo suficiente potentes y no existe un desarrollo análogo a What's Running para Linux. Falta una herramienta de integración que aporte sencillez y recoja en una sola aplicación información rutinaria para un análisis forense en vivo.

3. Gestión del proyecto

En este capítulo se habla sobre la planificación inicial calculada para la realización del proyecto. Por consiguiente, en función de la planificación inicial se detallará un presupuesto estimado para el proyecto.

3.1. Planificación

En este punto se va a detallar la planificación inicial desarrollada para la elaboración del proyecto. La planificación se ha dividido en tareas, las cuales abarcan desde el estudio de viabilidad del problema hasta la finalización del proyecto.

En la siguiente tabla se pueden apreciar las tareas definidas para la planificación, su duración en días, la fecha de comienzo y la fecha de finalización[11].

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**105 días**	**lun 03/01/11**	**vie 27/05/11**
Estudio de viabilidad	7 días	lun 03/01/11	mar 11/01/11
Documentación inicial	20 días	mié 12/01/11	mar 08/02/11
Análisis	13 días	mié 09/02/11	vie 25/02/11
Diseño	21 días	lun 28/02/11	lun 28/03/11
Implementación	36 días	mar 29/03/11	mar 17/05/11
Documentación del proyecto	28 días	mié 20/04/11	vie 27/05/11
Pruebas unitarias	24 días	vie 15/04/11	mié 18/05/11
Pruebas de sistema	7 días	jue 19/05/11	vie 27/05/11

Tabla 1: Tareas de la planificación del proyecto

Para este proyecto se ha contado con una única persona que ha ido asumiendo distintos roles en cada tarea del proyecto.

[11] Se ha establecido como inicio de proyecto el primer día laborable de enero de 2011, el lunes 3.

A continuación se presenta el diagrama de Gantt asociado a la planificación citada anteriormente.

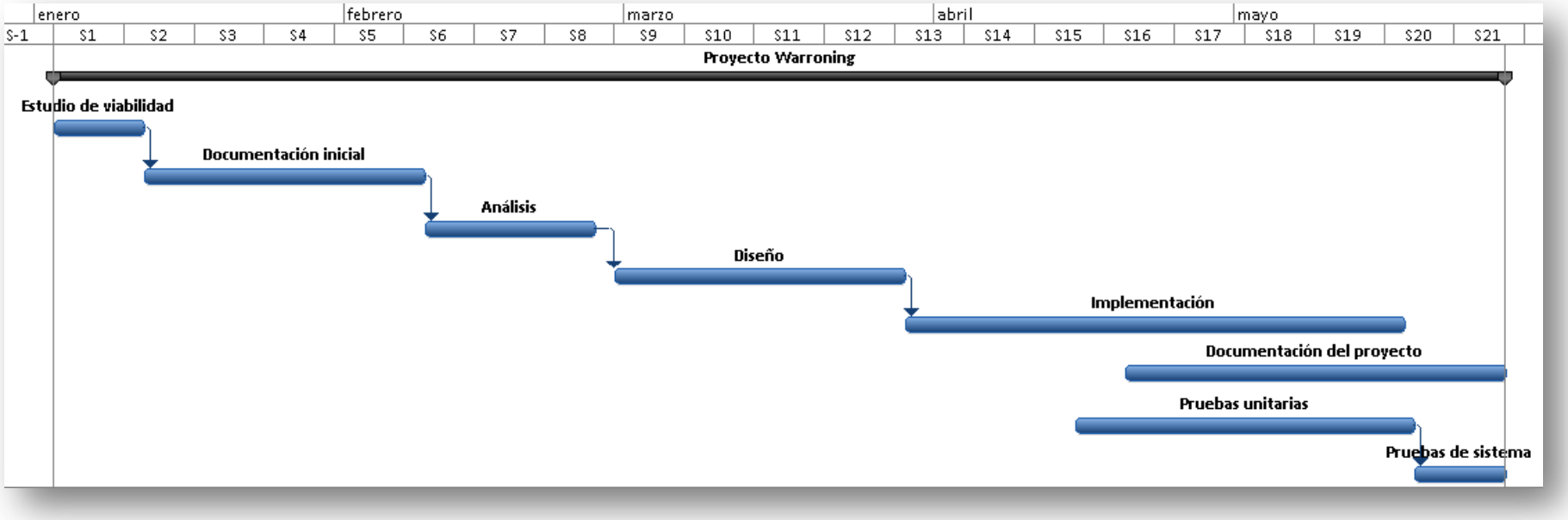


Ilustración 13: Diagrama de Gantt

Se ha estimado mediante la planificación inicial 105 días para concluir el proyecto. Se ha definido una dedicación de 4 horas diarias durante 22 días al mes. Finalmente se ha totalizado 420 horas para el desarrollo total del proyecto.

3.2. Presupuesto

En este apartado se detalla un presupuesto inicial calculado en función de la planificación del punto anterior. Se especifica gastos de personal, equipo y otros basándose en las plantillas de presupuesto para proyecto fin de carrera de la UC3M [4].

Como se comentó en la planificación, la misma persona ha ido asumiendo distintos roles. Los diferentes roles que han intervenido, así como su salario se pueden observar en la siguiente tabla.

Categoría profesional	Sueldo bruto / Año	Sueldo bruto / Mes	Coste empleado / Hora
Analista funcional	30.000 €	2.500 €	16 €
Analista orgánico	25.000 €	2.000 €	13 €
Jefe de proyecto	45.000 €	3.750 €	24 €
Programador	20.000 €	1.500 €	10 €
Técnico de pruebas	20.000 €	1.500 €	10 €

Tabla 2: Salario según categoría profesional

A continuación se presenta un desglose de las tareas en las que se indica la duración en horas de cada una, los recursos necesarios, el coste unitario y el coste total.

Tarea	Duración (h)	Recurso	Coste unitario (€)	Total (€)
Estudio de viabilidad	28	Jefe de proyecto (50%) Analista (50%)	24 16	28 * 0,5 * 24 = 336 28 * 0,5 * 16 = 224 **Total = 560**
Documentación inicial	80	Jefe de proyecto (70%) Analista (30%)	24 16	80 * 0,7 * 24 = 1.344 80 * 0,3 * 16 = 384 **Total = 1.728**
Análisis	52	Jefe de proyecto (10%) Analista funcional (90%)	24 16	52 * 0,1 * 24 = 124,8 52 * 0,9 * 16 = 748,8 **Total = 873,6**
Diseño	84	Jefe de proyecto (5%) Analista funcional (5%) Analista orgánico (90%)	24 16 13	84 * 0,05 * 24 = 100,8 84 * 0,05 * 16 = 67,2 84 * 0,9 * 13 = 982,8 **Total = 1.150,8**
Implementación	144	Jefe de proyecto (5%) Programador (95%)	24 10	144 * 0,05 * 24 = 172,8 144 * 0,95 * 10 = 1.368 **Total = 1.540,8**
Pruebas	124	Jefe de proyecto (5%) Técnico de pruebas (95%)	24 10	124 * 0,05 * 24 = 148,8 124 * 0,95 * 10 = 1.178 **Total = 1.326,8**
Documentación	112	Jefe de proyecto (5%) Analista funcional (5%) Programador (90%)	24 16 10	112 * 0,05 * 24 = 134,4 112 * 0,05 * 16 = 89,6 112 * 0,9 * 10 = 1.008 **Total = 1.232**
				Suma Totales = 8.412

Tabla 3: Coste de personal por tarea

Como resultado, los gastos de personal se elevan a OCHO MIL CUATROCIENTOS DOCE EUROS.

Intrínseco al personal involucrado en el desarrollo del proyecto se ha necesitado la utilización de distintas herramientas software y su respectivo hardware para ser utilizadas. Dichos conceptos se describen en la siguiente tabla.

Concepto	Coste (€)
Ordenador sobremesa	0
Sistema operativo Ubuntu 10.10	0
Sistema operativo Windows 7	0
Microsoft Office 2010	0
OpenOffice 3.2	0
NetBeans 6.9	0
Total	**0**

Tabla 4: Coste de hardware y software

El coste total dedicado a componentes hardware y software ha sido de 0 € debido a que el cliente del proyecto es la UC3M y ha cedido parte de su hardware y licencias software

para el desarrollo. Por otro lado, se ha usado software libre[12] para el 82 % del proyecto por lo que el coste en ese concepto ha sido 0 €.

En resumen, el presupuesto final del proyecto quedaría configurado como se muestra a continuación.

Concepto	Importe (€)
Personal	8.412
Hardware	0
Software	0
Costes indirectos (20%)	1.682,4
Total (sin IVA)	10.094,4
Total (IVA 18%)	**11.911,39**

Tabla 5: Presupuesto total del proyecto

Los costes indirectos de un proyecto software hacen referencia a dos grupos:

- Costes indirectos del proyecto: mantenimiento de sistemas informáticos, software de ayuda al desarrollo, herramientas CASE, etc.
- Costes indirectos generales de la empresa: personal administrativo, dirección, instalaciones y servicios generales, etc.

Como resultado, el presupuesto total del proyecto asciende a la cantidad de ONCE MIL NOVECIENTOS ONCE EUROS CON TREINTA Y NUEVE CÉNTIMOS para ser llevado a cabo en 105 días.

[12] Ubuntu, OpenOffice y NetBeans son herramientas de software libre. Su uso, distribución y desarrollo se hacen de forma gratuita.

4. Análisis

En este capítulo se va a tratar lo relativo a la fase de análisis del ciclo de desarrollo de software. Mediante reuniones con el cliente se han ido extrayendo necesidades que han sido transformadas en requisitos, inicialmente de usuario y finalmente de software.

Para la obtención de requisitos de usuario se ha usado una metodología basada en el estándar de la Agencia Espacial Europea (ESA) [5] porque el formato de las tablas era más reducido y manejable. En cambio, para la especificación de los requisitos de software se elegido Métrica V3 [6] porque realiza una catalogación más clara de los requisitos de software.

4.1. Requisitos de usuario

En este apartado se especifican los requisitos de usuario que se han extraído a raíz de las diversas reuniones con el cliente. Estos requisitos se clasifican en requisitos de capacidad (funcionales) y de restricción (no funcionales). Son la base de los requisitos de software que se describen en el siguiente apartado.

La ESA divide los requisitos de usuario en dos clases: requisitos de capacidad (RUC) y requisitos de restricción (RUR). Los RUC definen lo que la aplicación necesita hacer. Por otro lado, los RUR ponen límites o restricciones sobre lo que debe hacer.

Cada requisito de usuario se representará en una tabla que seguirá el siguiente formato:

- **Identificador:** código que identifica el requisito de usuario de manera unívoca. Sigue la regla de nombrado RUC-XX o RUR-XX, siendo RUC requisito de usuario de capacidad y RUR requisito de usuario de restricción. XX denota un valor numérico de dos cifras que empieza en 01 y se irá incrementando a modo de contador. Al llegar a los RUR se reinicia el contador.
- **Título:** campo que indica el título del requisito.
- **Descripción:** campo que describe detalladamente el requisito.
- **Prioridad:** campo que denota la prioridad del requisito dentro del proyecto. Puede tomar los siguientes valores: “baja”, “media”, “alta”.
- **Estabilidad:** campo que especifica si el requisito será susceptible de ser alterado o no a lo largo del desarrollo del proyecto. Puede tomar los siguientes valores: “estable” o “no estable”.

a) Requisitos de usuario de capacidad

Identificador	RUC-01
Título	Información de los procesos en ejecución
Descripción	El usuario podrá ver información de los procesos que se están ejecutando en la máquina en ese momento
Prioridad	Alta
Estabilidad	Estable

Tabla 6: RUC-01 Información de los procesos en ejecución

Identificador	RUC-02
Título	Información de las conexiones de red activas
Descripción	El usuario podrá ver información de las conexiones de red TCP/UDP activas en el sistema en ese momento
Prioridad	Alta
Estabilidad	Estable

Tabla 7: RUC-02 Información de las conexiones de red activas

Identificador	RUC-03
Título	Información de los módulos cargados en el kernel
Descripción	El usuario podrá ver información de los módulos cargados en el kernel del sistema en ese momento
Prioridad	Baja
Estabilidad	Estable

Tabla 8: RUC-03 Información de los módulos cargados en el kernel

Identificador	RUC-04
Título	Información de las conexiones SSH aceptadas
Descripción	El usuario podrá ver un histórico de las conexiones SSH que se han aceptado en la máquina
Prioridad	Media
Estabilidad	Estable

Tabla 9: RUC-04 Información de las conexiones SSH aceptadas

b) Requisitos de usuario de restricción

Identificador	RUR-01
Título	Aplicación portable
Descripción	La aplicación se debe poder ejecutar sin necesidad de una instalación previa
Prioridad	Alta
Estabilidad	Estable

Tabla 10: RUR-01 Aplicación portable

Identificador	RUR-02
Título	Aplicación autocontenida
Descripción	La aplicación no debe utilizar programas externos
Prioridad	Alta
Estabilidad	Estable

Tabla 11: RUR-02 Aplicación autocontenida

Identificador	RUR-03
Título	Ejecución en dispositivo de solo lectura
Descripción	La aplicación debe funcionar en un dispositivo de solo lectura
Prioridad	Alta
Estabilidad	Estable

Tabla 12: RUR-03 Ejecución en dispositivo de solo lectura

Identificador	RUR-04
Título	Compatible con Linux
Descripción	La aplicación debe funcionar en sistemas operativos de la familia Linux
Prioridad	Alta
Estabilidad	Estable

Tabla 13: RUR-04 Compatible con Linux

Identificador	RUR-05
Título	Ejecución como súper usuario
Descripción	La aplicación debe ser ejecutada como súper usuario
Prioridad	Alta
Estabilidad	Estable

Tabla 14: RUR-05 Ejecución como súper usuario

4.2. Requisitos de software

En este apartado se especifican los requisitos de software que se han extraído a raíz de los requisitos de usuario especificados en el punto anterior. Los requisitos de software son uno de los productos entrantes de la fase de diseño.

Siguiendo la metodología de desarrollo Métrica V3 [6], los requisitos de software se clasifican de la siguiente forma:

Tipo	Descripción
Funcionales	Definen las capacidades o funcionalidades
Disponibilidad del sistema	Definen los niveles de disponibilidad del software (cómo puede usarse el sistema cuando está operativo, su capacidad mínima y media disponible, etc.)
Implantación	Definen las características de entrega y puesta en marcha del software
Rendimiento	Establecen las limitaciones de funcionamiento del software (velocidad de las operaciones, uso de memoria, etc.)
Seguridad	Definen los criterios de seguridad para el software (restricciones de acceso, confidencialidad, integridad, etc.)

Tabla 15: Tipos de requisitos de software

Cada requisito de software se representará en una tabla que seguirá el siguiente formato:

- **Identificador:** código que identifica el requisito de software de manera unívoca. Sigue la regla de nombrado RSY-XX, siendo RSY un requisito de software del tipo Y, tomando la inicial de cada tipo de la tabla anterior (RSF, RSD, …). XX denota un valor numérico de dos cifras que empieza en 01 y se irá incrementando a modo de contador. Al cambiar de tipo se reinicia el contador.
- **Título:** campo que indica el título del requisito.
- **Descripción:** campo que describe detalladamente el requisito.
- **Prioridad:** campo que denota la prioridad del requisito dentro del proyecto. Puede tomar los siguientes valores: “baja”, “media”, “alta”.
- **Estabilidad:** campo que especifica si el requisito será susceptible de ser alterado o no a lo largo del desarrollo del proyecto. Puede tomar los siguientes valores: “estable” o “no estable”.

- **Prerrequisito:** si es necesario satisfacer otro requisito anterior a éste se especificará cuál en este campo.
- **Fuente:** indica la procedencia del requisito. En este caso se refiere al requisito de usuario del que procede.

a) Requisitos software funcionales:

Identificador	RSF-01
Título	Listado de procesos en ejecución
Descripción	La aplicación deberá mostrar una tabla con todos los procesos en ejecución en esa máquina
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUC-01

Tabla 16: RSF-01 Listado de procesos en ejecución

Identificador	RSF-02
Título	Detalle de un proceso en ejecución
Descripción	La aplicación deberá mostrar información detallada de cualquier proceso del listado de procesos en ejecución
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-01
Fuente	RUC-01

Tabla 17: RSF-02 Detalle de un proceso en ejecución

Identificador	RSF-03
Título	Resaltar procesos sospechosos
Descripción	Destacar en la tabla de procesos los procesos que estén teniendo un comportamiento sospechoso (que hayan elevado sus privilegios, usuario real distinto de usuario efectivo). En el detalle del proceso también debe resaltarse el campo que lo hace sospechoso
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-01, RSF-02
Fuente	RUC-01

Tabla 18: RSF-03: Resaltar procesos sospechosos

Identificador	RSF-04
Título	Listado de conexiones de red activas
Descripción	La aplicación debe mostrar una tabla con las conexiones activas en la máquina que se ejecuta bajo el protocolo TCP y UDP
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUC-02

Tabla 19: RSF-04 Listado de conexiones de red activas

Identificador	RSF-05
Título	Detalle de una conexión
Descripción	La aplicación debe mostrar información detallada de cualquier conexión del listado de conexiones de red activas
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-04
Fuente	RUC-02

Tabla 20: RSF-05 Detalle de una conexión

Identificador	RSF-06
Título	Navegación de conexión a proceso
Descripción	Desde una conexión de red se debe mostrar al usuario información del proceso que la usa, permitiendo la navegación a la tabla de procesos para conocer su detalle
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-01, RSF-02, RSF-04, RSF-05
Fuente	RUC-02

Tabla 21: RSF-06 Navegación de conexión a proceso

Identificador	RSF-07
Título	Resaltar conexiones sospechosas
Descripción	Destacar las conexiones que han sido lanzadas por procesos que estén teniendo un comportamiento sospechoso (que hayan elevado sus privilegios, usuario real distinto al usuario efectivo)
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-03, RSF-04, RSF-05
Fuente	RUC-02

Tabla 22: RSF-07 Resaltar conexiones sospechosas

Identificador	RSF-08
Título	Listado de los módulos
Descripción	La aplicación debe mostrar una tabla con los módulos cargados en el kernel en ese momento
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUC-03

Tabla 23: RSF-08 Listado de los módulos

Identificador	RSF-09
Título	Detalle de un módulo
Descripción	La aplicación debe mostrar información detallada de un módulo cargado en el kernel en ese momento
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-08
Fuente	RUC-03

Identificador	RSF-10
Título	Listado de conexiones SSH aceptadas
Descripción	La aplicación debe mostrar un listado con el histórico de las conexiones SSH aceptadas por la máquina
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUC-04

Tabla 24: RSF-10 Listado de conexiones SSH aceptadas

Identificador	RSF-11
Título	Detalle de una conexión SSH
Descripción	Para cada conexión del listado de conexiones SSH se debe mostrar información adicional de la conexión como: fecha y hora de la conexión, dirección ip de la máquina que la hizo, usuario con el que se accedió
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-10
Fuente	RUC-04

Tabla 25: RSF-11 Detalle de una conexión SSH

Identificador	RSF-12
Título	Histórico de comandos
Descripción	Desde el listado de conexiones SSH aceptadas se podrá consultar los comandos que ha introducido el usuario en la máquina contenidos en el fichero .bash_history de su carpeta personal (home)
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	RSF-10, RSF-11
Fuente	RUC-04

Tabla 26: RSF-12 Histórico de comandos

b) Requisitos de software de disponibilidad:

Identificador	RSD-01
Título	Sistema de ficheros Unix
Descripción	La aplicación debe ser ejecutada sobre una máquina con un sistema de ficheros basado en Unix
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-04

Tabla 27: RSD-01 Sistema de ficheros Unix

Identificador	RSD-02
Título	Sistema operativo Linux
Descripción	La aplicación debe ser ejecutada sobre una máquina con sistema operativo Linux
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-04

Tabla 28: RSD-02 Sistema operativo Linux

Identificador	RSD-03
Título	Entorno gráfico
Descripción	La aplicación debe ser ejecutada sobre un sistema operativo que posea entorno gráfico
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-04

Tabla 29: RSD-03 Entorno gráfico

c) Requisitos de software de implantación:

No procede debido a que la aplicación desarrollada en este proyecto no consta de fase de implantación.

d) Requisitos de software de rendimiento:

Identificador	RSR-01
Título	Aplicación desarrollada en Java
Descripción	La aplicación debe ser desarrollada en el lenguaje de programación Java
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-01

Tabla 30: RSR-01 Aplicación desarrollada en Java

Identificador	RSR-02
Título	JRE incluido en la aplicación
Descripción	La aplicación debe incluir el conjunto de herramientas JRE (Java Run Time), necesario para ejecutar la aplicación
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-01

Tabla 31: RSR-02 JRE incluido en la aplicación

Identificador	RSR-03
Título	Ejecución desde dispositivo de solo lectura
Descripción	La aplicación se debe poder ejecutar únicamente desde un dispositivo de solo lectura
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-03

Tabla 32: RSR-03 Ejecución desde dispositivo de solo lectura

Identificador	RSR-04
Título	No escribir datos en el sistema
Descripción	La aplicación no debe escribir datos en la máquina que está siendo ejecutada
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-03

Tabla 33: RSR-04 No escribir datos en el sistema

e) Requisitos de software de seguridad:

Identificador	RSS-01
Título	Información obtenida del kernel
Descripción	La información que maneje la aplicación debe ser obtenida directamente de las tablas del kernel y no de programas intermedios
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-02

Tabla 34: RSS-01 Información obtenida del kernel

Identificador	RSS-02
Título	No usar programas del sistema
Descripción	La aplicación no debe utilizar bajo ningún concepto programas del sistema
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-02

Tabla 35: RSS-02 No usar programas del sistema

Identificador	RSS-03
Título	Ejecución como root
Descripción	La aplicación se debe ejecutar como usuario root para poder acceder a todos ficheros del sistema
Prioridad	Alta
Estabilidad	Estable
Prerrequisito	Ninguno
Fuente	RUR-05

Tabla 36: RSS-03 Ejecución como root

4.3. Matriz de trazabilidad

A continuación se incluye una matriz de trazabilidad que relaciona los requisitos de usuario con los requisitos de software, de esta forma se puede saber rápidamente si todos los requisitos de usuario han sido cubiertos.

	RUC-01	RUC-02	RUC-03	RUC-04	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05
RSF-01	X								
RSF-02	X								
RSF-03	X								
RSF-04		X							
RSF-05		X							
RSF-06		X							
RSF-07		X							
RSF-08			X						
RSF-09			X						
RSF-10				X					
RSF-11				X					
RSF-12				X					
RSD-01								X	
RSD-02								X	
RSD-03								X	
RSR-01					X				
RSR-02					X				
RSR-03							X		
RSR-04							X		
RSS-01						X			
RSS-02						X			
RSS-03									X

Tabla 37: Matriz de trazabilidad

5. Diseño

En este capítulo se va a tratar lo relativo a la fase de diseño del ciclo de desarrollo de software. En concreto se va a detallar lo relativo al diseño arquitectónico y el diseño de la interfaz de usuario.

5.1. Diseño arquitectónico

La aplicación ha sido pensada para ser ejecutada en local. El usuario debe transportar la aplicación en un dispositivo externo de sólo lectura e insertarlo en la máquina que quiere analizar. Para la fase de diseño se ha tenido en cuenta que la aplicación debe ser autocontenida y portable.

El diseño de la arquitectura software se ha basado en el patrón Modelo Vista Controlador (MVC de ahora de adelante). Este patrón se separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

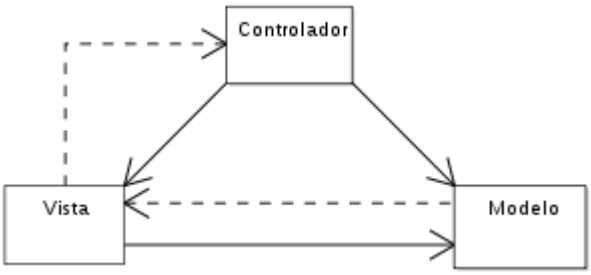


Ilustración 14: Diagrama de relación entre el modelo, la vista y el controlador

Cada componente del patrón se puede describir como sigue:

- **Modelo:** es la representación específica de la información con la cual el sistema opera. El modelo recibe las peticiones del controlador, las procesa y envía los datos a la vista para que los presente al usuario.
- **Vista:** presenta la información procedente del modelo en un formato adecuado para interactuar. En este caso es la interfaz de usuario.
- **Controlador:** responde a eventos, desencadenados por el usuario desde la vista a través de acciones e invoca peticiones al modelo y, probablemente, a la vista.

Gracias al MVC se separa la capa visual de su correspondiente programación y acceso a datos, mejorando el desarrollo y el mantenimiento.

A su vez, el modelo se divide en varios componentes para mantener un desarrollo escalable y modular.

En la siguiente ilustración se muestran los principales componentes de la aplicación desarrollada así como las relaciones existentes entre ellos mismos.

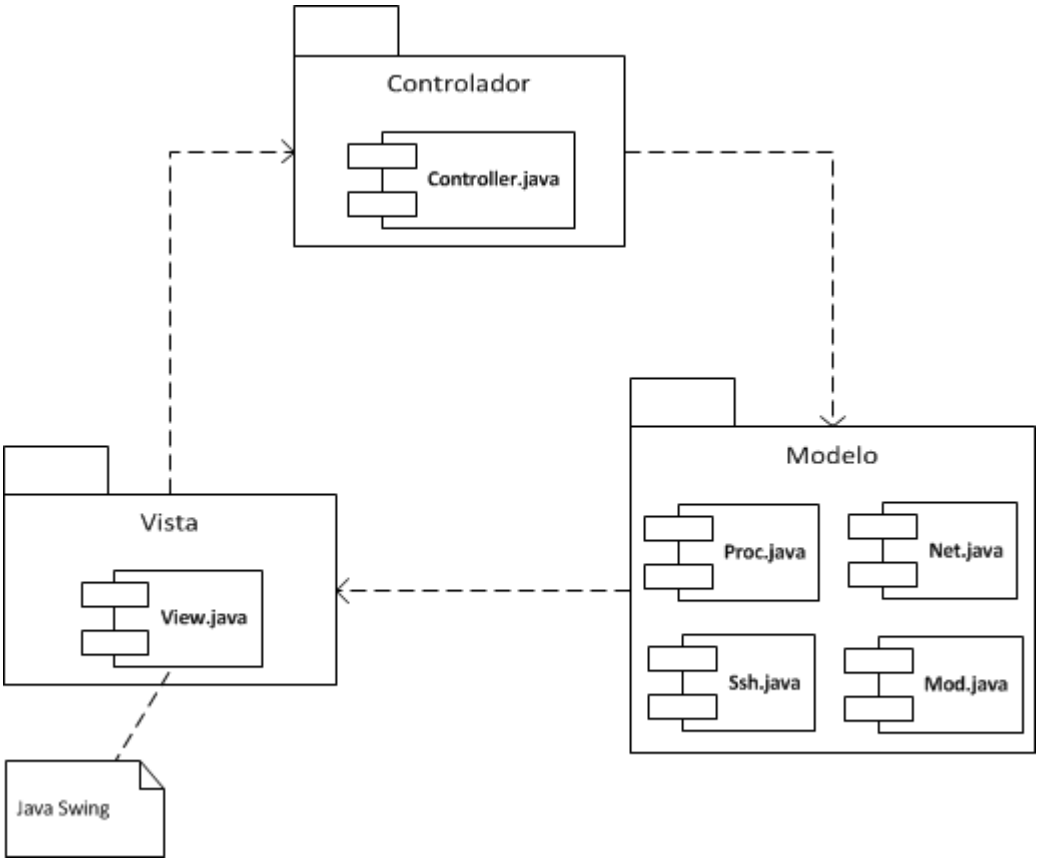


Ilustración 15: Diagrama de componentes

A continuación se va a describir cada uno de los componentes del diagrama anterior siguiendo un formato tabular que tendrá la siguiente estructura:

- **Identificador:** identifica de forma unívoca el componente. El código de identificador sigue el formato COM-XX, siendo XX un número entero positivo autoincremental.
- **Título:** el nombre que figura en el diagrama para el componente.
- **Finalidad:** indica el objetivo del componente.
- **Funcionalidades:** especifica las funciones que desempeña el componente dentro del sistema.

Identificador	COM-01
Título	View.java
Finalidad	Es el componente que gestiona la vista. Conforma la interfaz de usuario. Permite que el usuario interactúe con la aplicación. Mediante acciones se comunica con el componente controlador
Funcionalidades	• Presenta agrupa la información en pestañas • Gestiona las llamadas al controlador • Ofrece un menú de opciones • Presenta un botón de refresco para actualizar los datos visualizados

Tabla 38: COM-01 View.java

Identificador	COM-02
Título	Controller.java
Finalidad	Es el componente que gestiona el controlador. Es una capa intermedia que separa la vista del modelo.
Funcionalidades	• Recibe acciones de la vista • Invoca métodos específicos del modelo para cada acción

Tabla 39: COM-02 Controller.java

Identificador	COM-03
Título	Proc.java
Finalidad	Es la parte del modelo dedicada a gestionar la información relativa a los procesos del sistema
Funcionalidades	• Lista los procesos que se están ejecutando • Lee el estado de los procesos • Envía datos a la vista

Tabla 40: COM-03 Proc.java

Identificador	COM-04
Título	Net.java
Finalidad	Es la parte del modelo dedicada a gestionar la información relativa a las conexiones activas de red del sistema
Funcionalidades	• Lista las conexiones de red que se encuentran activas en el sistema • Lee el estado de dichas conexiones • Envía datos a la vista

Tabla 41: COM-04 Net.java

Identificador	COM-05
Título	Ssh.java
Finalidad	Es la parte del modelo dedicada a gestionar la información relativa a las conexiones SSH que ha recibido la máquina
Funcionalidades	• Lista los accesos SSH que ha aceptado el sistema • Muestra el histórico de comandos introducidos por un usuario • Envía datos a la vista

Tabla 42: COM-05 Ssh.java

Identificador	COM-06
Título	Mod.java
Finalidad	Es la parte del modelo dedicada a gestionar la información relativa a los módulos cargados en el kernel
Funcionalidades	• Lista los módulos cargamos por el kernel (LKM) • Lee el estado de cada módulo • Envía datos a la vista

Tabla 43: COM-06 Mod.java

5.2. Diseño de la interfaz de usuario

En esta sección se van a mostrar las distintas interfaces que se han diseñado para la aplicación, cómo el usuario puede interactuar con las mismas y por último qué resultado puede obtener con sus acciones.

5.2.1. Detalles generales

De forma general, la estructura de la interfaz se desglosa en una parte de opciones generales situada en el margen superior de la ventana y un panel de pestañas que presenta la información. Cada pestaña contiene un módulo. Se han desarrollado los correspondientes a procesos, conexiones de red, módulos cargados en el kernel y conexiones SSH aceptadas.

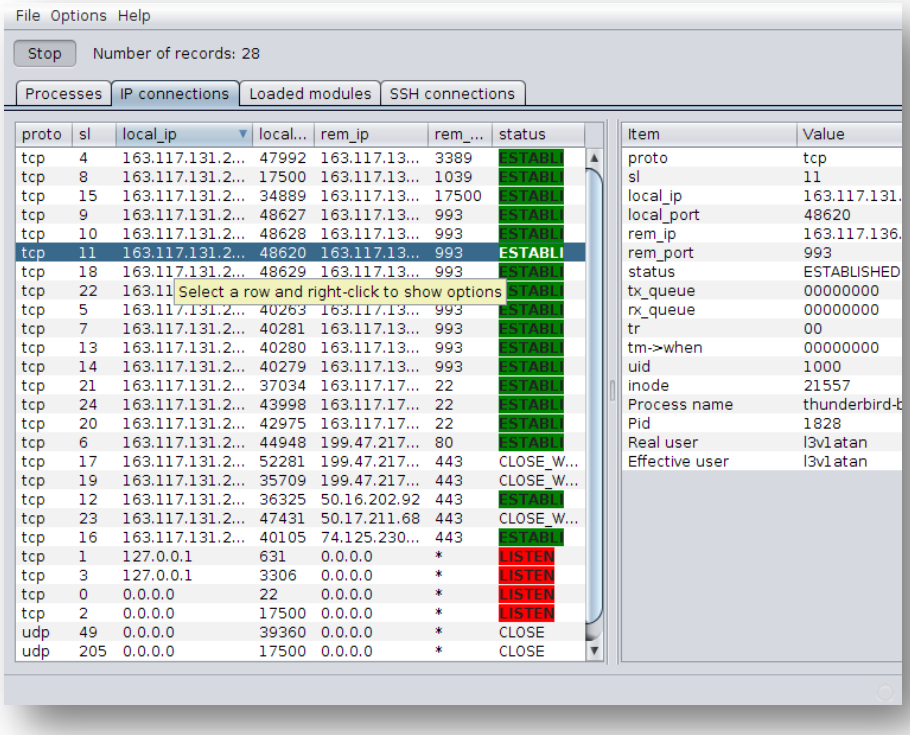


Ilustración 16: IU Vista general

De forma más detallada, la aplicación está compuesta por los siguientes componentes:

- **Menú principal:** se encuentra en la parte superior de la ventana. Está agrupado en tres desplegables: desde File podemos salir de la aplicación con la opción Exit; en Options podemos generar un XML global con los datos contenidos en todas los módulos; en Help se presenta información variada en el apartado About.

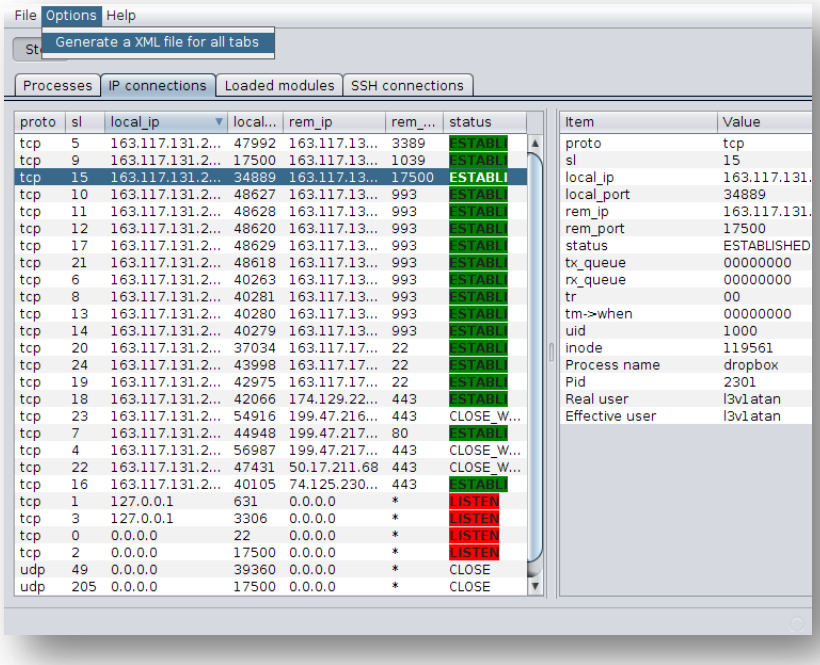


Ilustración 17: IU Menú principal. Generar XML global

- **Botón start/stop:** habilita o deshabilita el autorefresco de los datos mostrados en las pestañas.
- **Etiqueta "Number of records":** indica el número de entradas que hay en la tabla que se está mostrando.
- **Panel de pestañas:** cada sección tiene asignado un módulo de la aplicación y se explicarán con detalle en los siguientes puntos. Para cada uno se puede generar un fichero XML pinchando con el botón derecho sobre el listado y seleccionando la opción "Save a XML file for this tab".

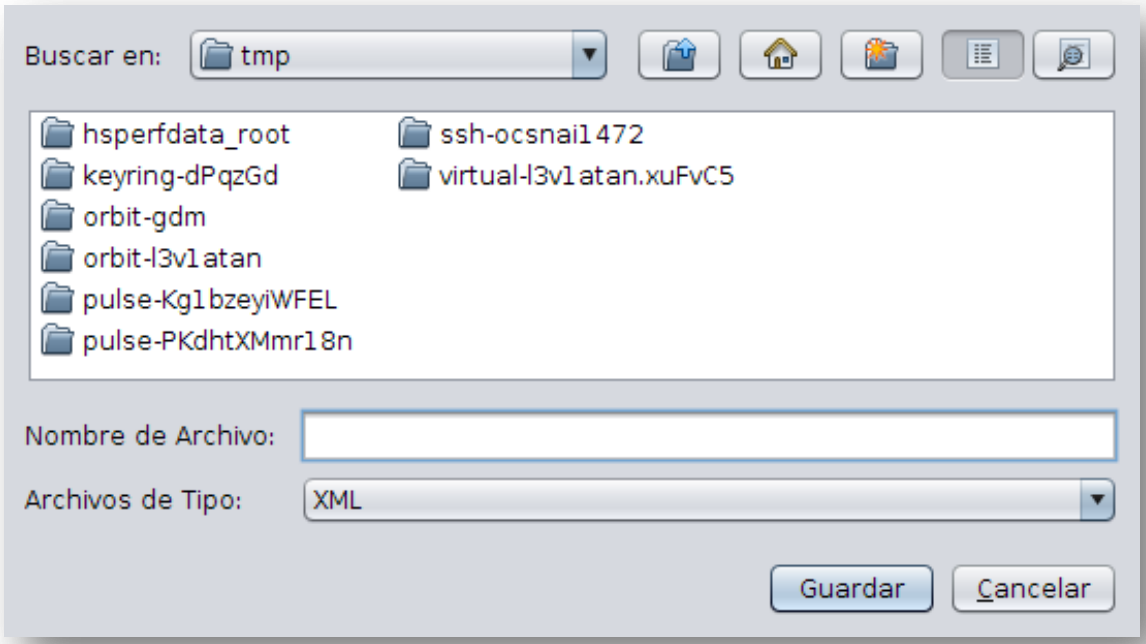


Ilustración 18: IU Diálogo para salvar un fichero XML

5.2.2. Procesos

Desde esta sección obtiene toda la información sobre los procesos que se encuentran en memoria.

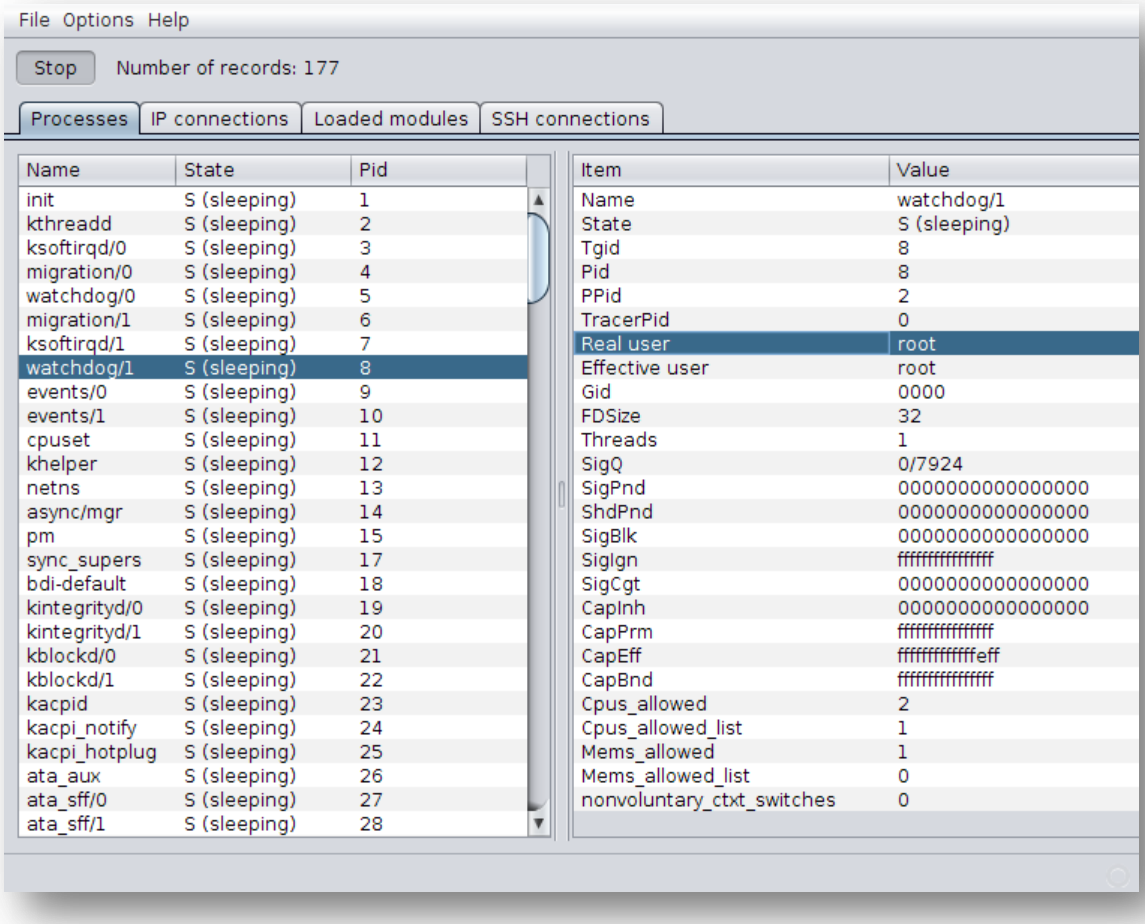


Ilustración 19: IU Procesos

Esta vista consta de dos tablas. En la izquierda podemos ver un listado de procesos en los que se especifica su nombre, su estado y su identificador de proceso (columnas Name, State y PID[13], respectivamente). En la derecha se encuentra la información detallada del proceso seleccionado del listado anterior mediante el par elemento-valor (columnas Item y Value).

Estos datos se obtienen, para cada proceso, del fichero /proc/[PID]/status, siendo [PID] el identificador del proceso. Un ejemplo de fichero status de un proceso podría ser el siguiente:

[13] PID: process identificator, es un número entero positivo que identifica a un proceso de forma unívoca en el kernel.

```
$ cat /proc/3515/status

Name:   bash
State:  S (sleeping)
Tgid:   3515
Pid:    3515
PPid:   3452
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    100     100     100     100
FDSize: 256
Groups: 16 33 100
VmPeak:     9136 kB
VmSize:     7896 kB
VmLck:         0 kB
VmHWM:      7572 kB
VmRSS:      6316 kB
VmData:     5224 kB
VmStk:        88 kB
VmExe:       572 kB
VmLib:      1708 kB
VmPTE:        20 kB
Threads:        1
SigQ:   0/3067
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000010000
SigIgn: 0000000000384004
SigCgt: 000000004b813efb
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed:   00000001
Cpus_allowed_list:      0
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        150
nonvoluntary ctxt switches:     545
```

Ilustración 20: Ejemplo /proc/3515/status

Se pueden destacar los siguientes campos:

- Name: comando arrancado por este proceso
- State: estado actual del proceso. Toma los valores "R (running)", "S (sleeping)", "D (disk sleep)", "T (stopped)", "T (tracing stop)", "Z (zombie)", o "X (dead)".
- Pid: identificador del proceso.
- Uid: identificador de usuario real, efectivo, saved set y sistema de ficheros.

Respecto al UID, el usuario real corresponde al usuario que ha invocado al proceso. El usuario efectivo indica los privilegios con los que se está ejecutando un proceso. Puede ser distinto al usuario real. Por ejemplo el comando para cambiar la contraseña de usuario,

passwd, tiene siempre como usuario efectivo root porque necesita modificar el fichero /etc/shadow, pero su usuario real es el que invocó el comando. Si se detecta un cambio entre el usuario real y el efectivo se resaltará en rojo el nombre del proceso porque podría ser debido a una escala de privilegios no autorizada.

Para obtener información más detallada consulte el anexo I, representación de procesos en el kernel.

5.2.3. Conexiones de red

En esta pestaña se muestra la información sobre las conexiones de red que se encuentran activas.

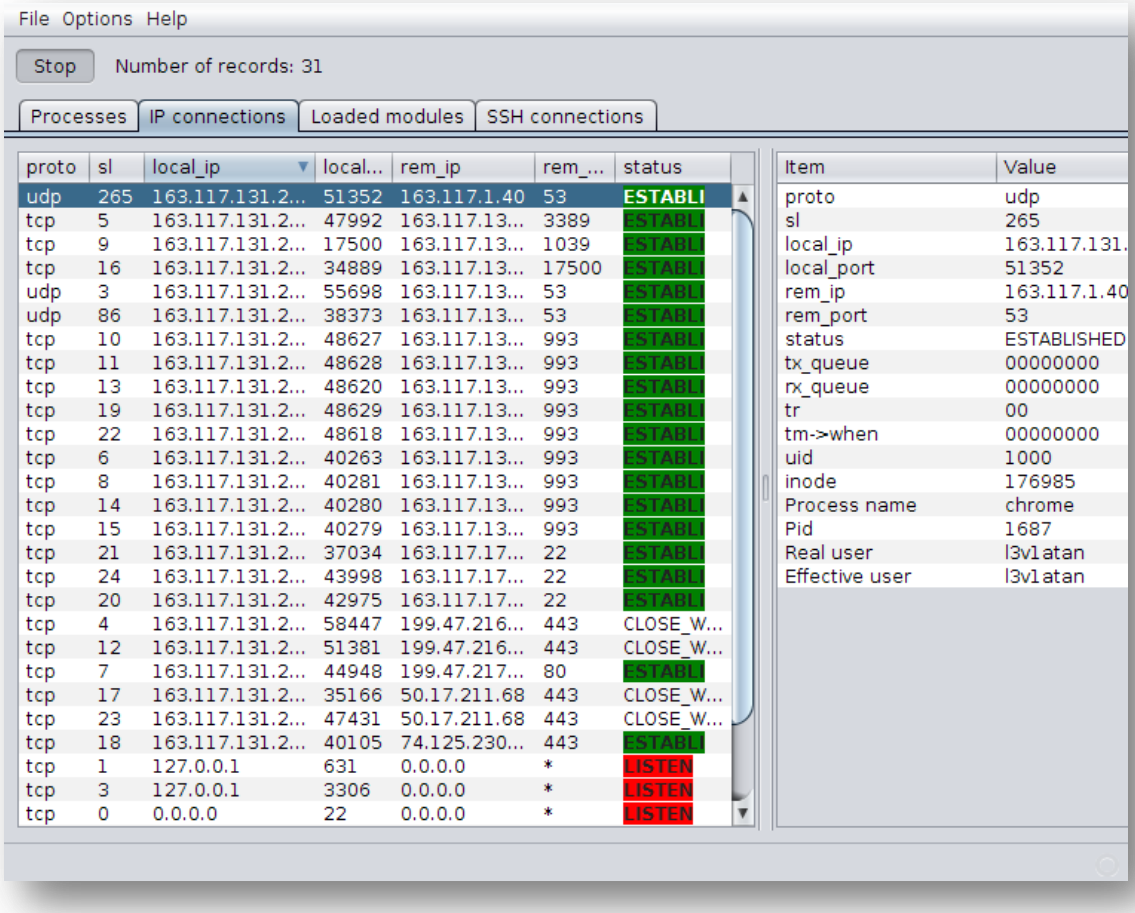


Ilustración 21: IU Conexiones de red

Esta vista consta de dos tablas. En la izquierda podemos ver un listado de las conexiones de red con información general, entre las que se pueden destacar las direcciones IP y puertos involucrados, estado de la conexión, protocolo usado, etc. En la derecha se encuentra la información detallada del proceso seleccionado del listado anterior mediante el par elemento-valor (columnas Item y Value).

Estos datos se obtienen de los ficheros /proc/net/tcp y /proc/net/udp, para sockets[14] TCP y UDP respectivamente. A continuación se pueden observar unos fragmentos de ejemplo de cada fichero:

```
$ cat /proc/net/tcp

  sl  local_address rem_address   st tx_queue rx_queue tr tm-
>when retrnsmt   uid  timeout inode
   0: 00000000:0050 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 7750 1 f35b4a00 300 0 0 2 -1
   1: 00000000:0016 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 6782 1 f35b4000 300 0 0 2 -1
   2: 0100007F:0277 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 2937355 1 e1678000 300 0 0 2 -1
   3: 00000000:445C 00000000:0000 0A 00000000:00000000
00:00000000 00000000  1000        0 299996 1 f2d26800 300 0 0 2 -1
   4: 0100007F:0CEA 00000000:0000 0A 00000000:00000000
00:00000000 00000000   116        0 8400 1 f35b4f00 300 0 0 2 -1
   5: D18375A3:DC26 72C381AE:01BB 08 00000000:00000026
00:00000000 00000000  1000        0 1187067 1 e167a300 36 4 28 3 2
```

Ilustración 22: Ejemplo /proc/net/tcp

```
$ cat /proc/net/udp

  sl  local_address rem_address   st tx_queue rx_queue tr tm-
>when retrnsmt   uid  timeout inode ref pointer drops
  30: 00000000:89AD 00000000:0000 07 00000000:00000000
00:00000000 00000000   104        0 6912 2 f3d88280 0
 205: 00000000:445C 00000000:0000 07 00000000:00000000
00:00000000 00000000  1000        0 299992 2 f3d89180 0
 346: 00000000:14E9 00000000:0000 07 00000000:00000000
00:00000000 00000000   104        0 6910 2 f3d88000 0
```

Ilustración 23: Ejemplo /proc/net/udp

Resulta interesante resaltar los siguientes campos:

- sl: número de la línea en el listado de salida.
- local_address: dirección IP local y número de puerto para el socket. La dirección IP se muestra en 4 bytes hexadecimales en formato little-endian[15]. El número de puerto es un simple par de bytes en hexadecimal. A continuación se muestra el algoritmo seguido para traducir la dirección IP:

[14] Un socket es un concepto abstracto por el cual dos programas, probablemente ubicados en equipos distintos, establecen una conexión por la que pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

[15] Little-endian es un formato de codificación en el que el byte menos significativo es el primero por lo que es necesario invertir el orden de los bytes para facilitar su compresión.

1. Cogemos el dato de entrada: C801A8C0:BDF2.
2. Como la dirección está en formato little-endian hay que darle la vuelta: C0A801C8:BDF2.
3. Separamos por puntos: C0.A8.01.C8:BDF2.
4. Se traducen los valores hexadecimales: 192.168.1.200:48626.

- rem_address: dirección IP remota y el número de puerto para el socket. La codificación es la misma de antes.
- st: estado del socket. Este campo toma un valor hexadecimal definido en el siguiente enumerado[16]:

```
STATUS      -     HEX VALUE
ESTABLISHED -     01
SYN_SENT    -     02
SYN_RECV    -     03
FIN_WAIT1   -     04
FIN_WAIT2   -     05
TIME_WAIT   -     06
CLOSE       -     07
CLOSE_WAIT  -     08
LAST_ACK    -     09
```

Ilustración 24: Correspondencia estado - valor hexadecimal de un socket

En el anexo II, representación de las conexiones de red en el kernel puede consultar información detallada sobre los datos que se muestran en la vista de conexiones de red.

Existe la posibilidad de ver información del proceso que está usando la conexión. En el detalle se pueden consultar los campos Process name, Pid, Real user y Effective user. Aprovechando esa información se ha implementado la posibilidad de realizar una navegación de la conexión de la pestaña Net al proceso de la pestaña Proc. Si accedemos al menú emergente después de seleccionar una entrada de la tabla se muestra la opción “Navigate to process” con la que se puede realizar la acción.

[16] Definido en la interfaz include/net/tcp_states.h del kernel de Linux desde la versión 2.6. En versiones anteriores como 2.4 se puede encontrar en include/Linux/tcp.h. Consultado en *TOMOYO Linux: Cross reference of source code*

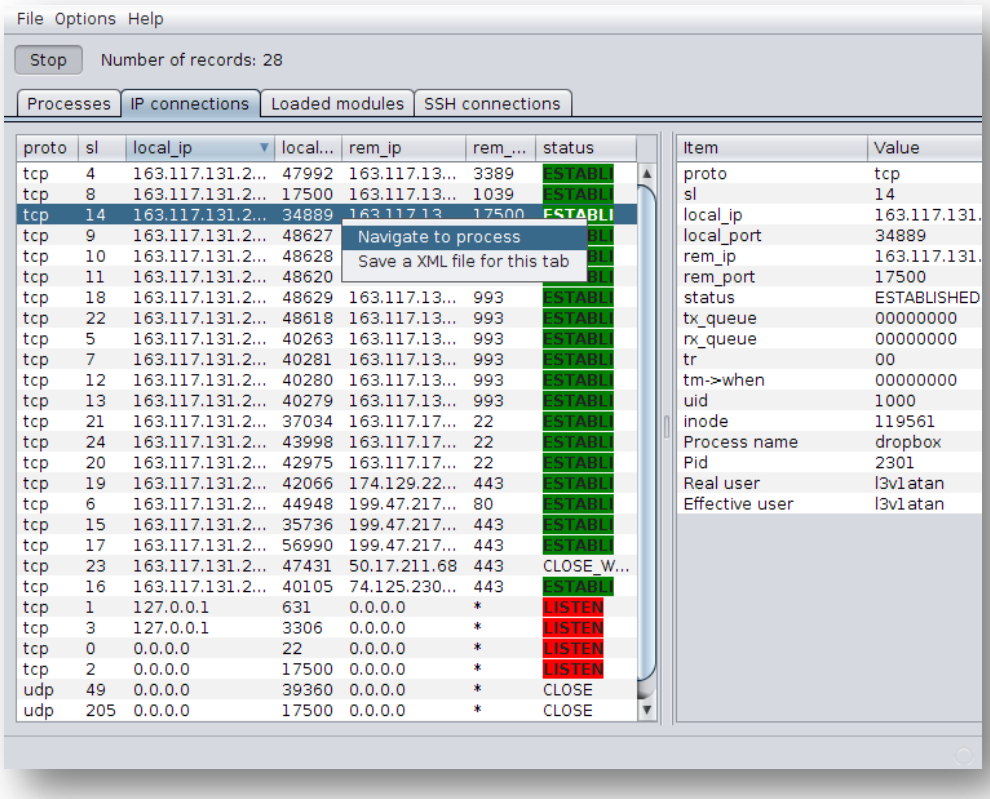
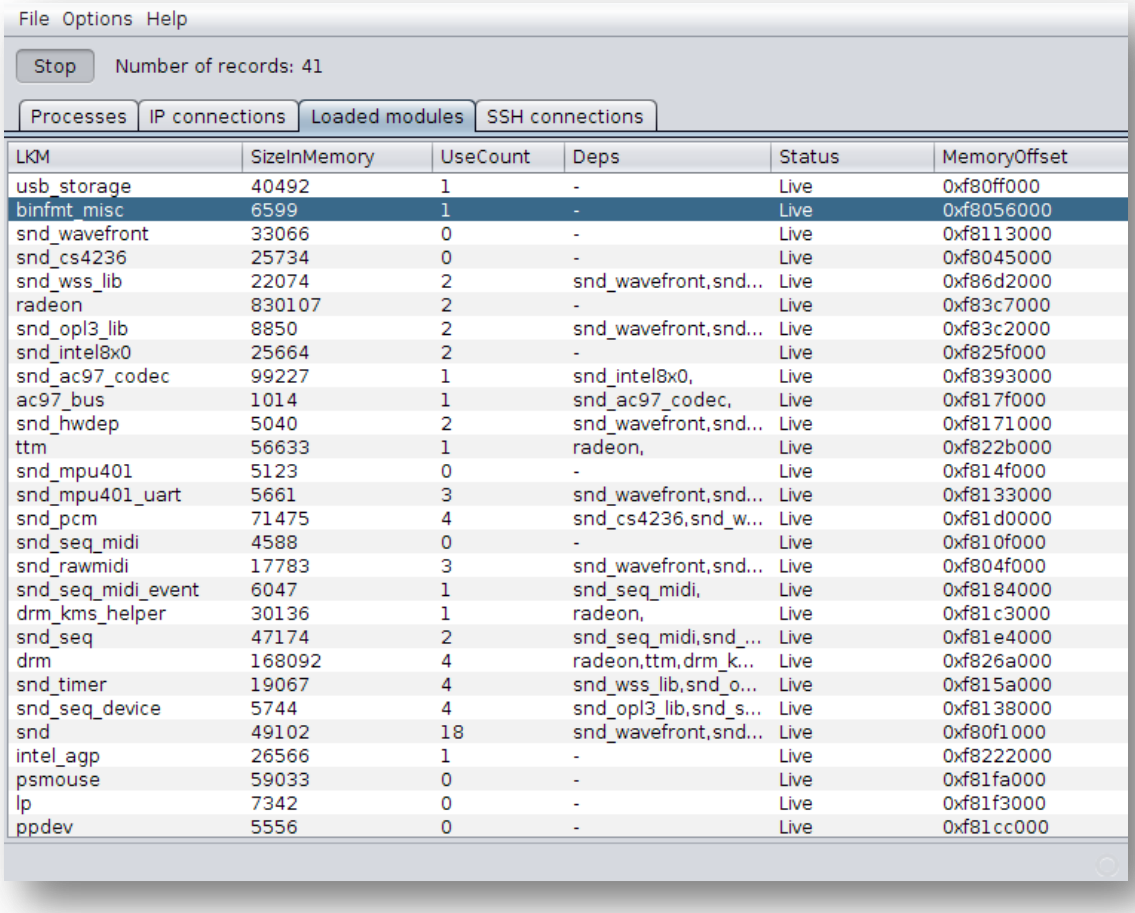


Ilustración 25: IU Popup menu módulo de conexiones de red

5.2.4. Módulos cargados en el kernel

Desde este apartado se obtiene toda la información relativa a los módulos cargados en el kernel (Linux Kernel Modules, LKM).



LKM	SizeInMemory	UseCount	Deps	Status	MemoryOffset
usb_storage	40492	1	-	Live	0xf80ff000
binfmt_misc	6599	1	-	Live	0xf8056000
snd_wavefront	33066	0	-	Live	0xf8113000
snd_cs4236	25734	0	-	Live	0xf8045000
snd_wss_lib	22074	2	snd_wavefront,snd...	Live	0xf86d2000
radeon	830107	2	-	Live	0xf83c7000
snd_opl3_lib	8850	2	snd_wavefront,snd...	Live	0xf83c2000
snd_intel8x0	25664	2	-	Live	0xf825f000
snd_ac97_codec	99227	1	snd_intel8x0,	Live	0xf8393000
ac97_bus	1014	1	snd_ac97_codec,	Live	0xf817f000
snd_hwdep	5040	2	snd_wavefront,snd...	Live	0xf8171000
ttm	56633	1	radeon,	Live	0xf822b000
snd_mpu401	5123	0	-	Live	0xf814f000
snd_mpu401_uart	5661	3	snd_wavefront,snd...	Live	0xf8133000
snd_pcm	71475	4	snd_cs4236,snd_w...	Live	0xf81d0000
snd_seq_midi	4588	0	-	Live	0xf810f000
snd_rawmidi	17783	3	snd_wavefront,snd...	Live	0xf804f000
snd_seq_midi_event	6047	1	snd_seq_midi,	Live	0xf8184000
drm_kms_helper	30136	1	radeon,	Live	0xf81c3000
snd_seq	47174	2	snd_seq_midi,snd_...	Live	0xf81e4000
drm	168092	4	radeon,ttm,drm_k...	Live	0xf826a000
snd_timer	19067	4	snd_wss_lib,snd_o...	Live	0xf815a000
snd_seq_device	5744	4	snd_opl3_lib,snd_s...	Live	0xf8138000
snd	49102	18	snd_wavefront,snd...	Live	0xf80f1000
intel_agp	26566	1	-	Live	0xf8222000
psmouse	59033	0	-	Live	0xf81fa000
lp	7342	0	-	Live	0xf81f3000
ppdev	5556	0	-	Live	0xf81cc000

Ilustración 26: IU Módulos cargados

Los datos se muestran en una única tabla en la que se puede ver un listado de módulos en el que para cada uno se especifica: nombre del módulo, tamaño que ocupan en memoria, número de instancias del módulo cargadas, módulos de los que depende, estado y offset[17] de memoria (columnas LKM, SizeInMemory, UseCount, Deps, Status y MemoryOffset, respectivamente).

La información mostrada proviene del fichero /proc/modules. A continuación se muestra un ejemplo:

[17] El término offset, en informática, significa desplazamiento. Si el espacio de memoria del kernel comienza en la dirección X, el módulo estará en X+OFFSET.

```
$ cat /proc/modules

nfs      170109  0 -          Live 0x129b0000
lockd    51593   1 nfs,       Live 0x128b0000
nls_utf8 1729    0 -          Live 0x12830000
vfat     12097   0 -          Live 0x12823000
fat      38881   1 vfat,      Live 0x1287b000
autofs4  20293   2 -          Live 0x1284f000
sunrpc   140453  3 nfs,lockd, Live 0x12954000
3c59x    33257   0 -          Live 0x12871000
uhci_hcd 28377   0 -          Live 0x12869000
md5      3777    1 -          Live 0x1282c000
ipv6     211845 16 -          Live 0x128de000
ext3     92585   2 -          Live 0x12886000
jbd      65625   1 ext3,      Live 0x12857000
dm mod   46677   3 -          Live 0x12833000
```

Ilustración 27: Ejemplo /proc/modules

En el anexo III, representación de módulos cargados en el kernel, se explica en detalle la información representada en cada columna.

5.2.5. Conexiones establecidas vía SSH

Este módulo se encarga de mostrar la información relativa a las conexiones aceptadas vía SSH.

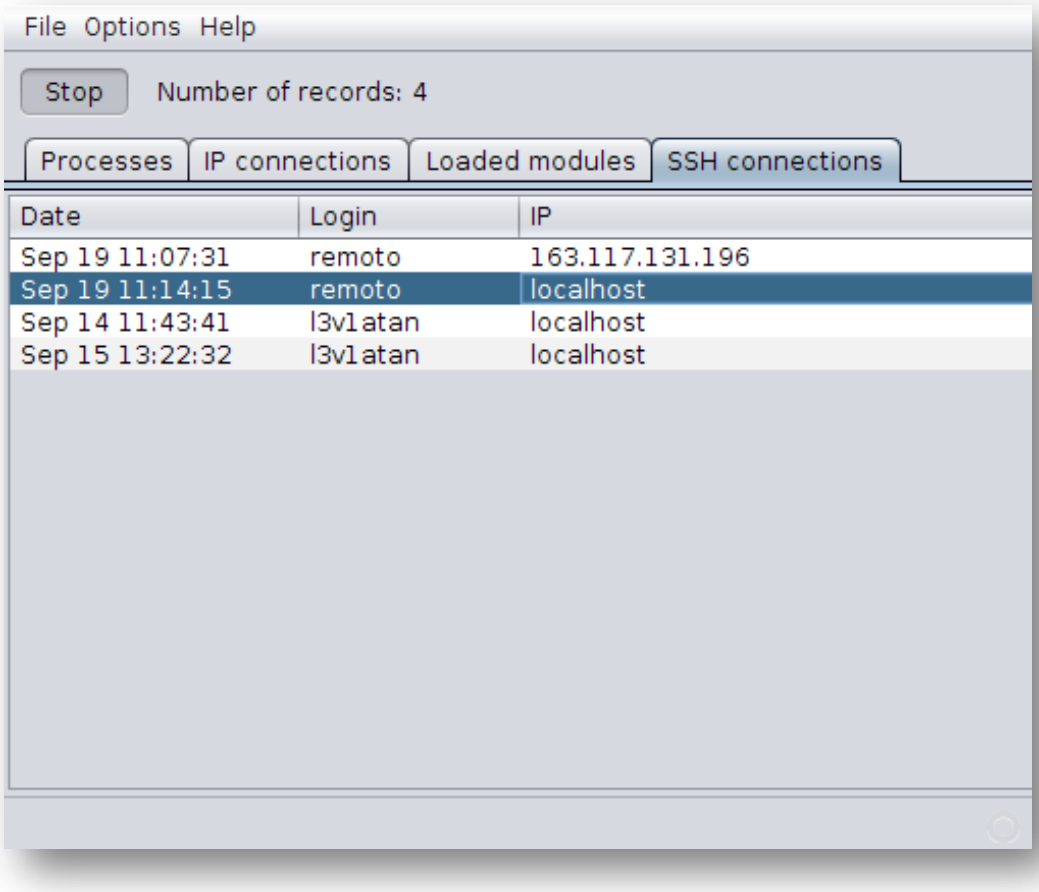


Ilustración 28: IU Conexiones establecidas vía SSH

Por cada fila se encuentras los siguientes datos:

- **Date:** fecha en la que se realizó la conexión.
- **Login:** login con el que se accedió.
- **IP:** dirección IP desde la que se conectó.

Esta información se obtiene del fichero de log de autorizaciones del sistema, /var/log/auth.log[18]. A continuación se muestra un fragmento del fichero:

[18] En Linux Debian/Ubuntu. En otras distribuciones puede cambiar. Por ejemplo, Red Hat los almacena en /var/log/secure.

```
      $ tail /var/log/auth.log

      Aug 10 10:39:01 sissomah CRON[6973]: pam_unix(cron:session):
session opened for user root by (uid=0)
      Aug 10 10:39:01 sissomah CRON[6973]: pam_unix(cron:session):
session closed for user root
      Aug 10 11:09:01 sissomah CRON[7126]: pam_unix(cron:session):
session opened for user root by (uid=0)
      Aug 10 11:09:01 sissomah CRON[7126]: pam_unix(cron:session):
session closed for user root
      Aug 10 11:17:01 sissomah CRON[7135]: pam_unix(cron:session):
session opened for user root by (uid=0)
      Aug 10 11:17:01 sissomah CRON[7135]: pam_unix(cron:session):
session closed for user root
      Aug 10 11:20:10 sissomah sshd[7141]: Accepted password for
jmlotero from ::1 port 42890 ssh2
```

Ilustración 29: Ejemplo /var/log/auth.log

Puede consultar información detallada sobre las conexiones SSH en Linux en el anexo V.

Si se pulsa con el botón derecho sobre una entrada aparecerá un menú emergente que mostrará los elementos “Save a XML file for this tab” (explicado en el apartado 5.2.1 Detalles generales) y “Show .bash_history”.

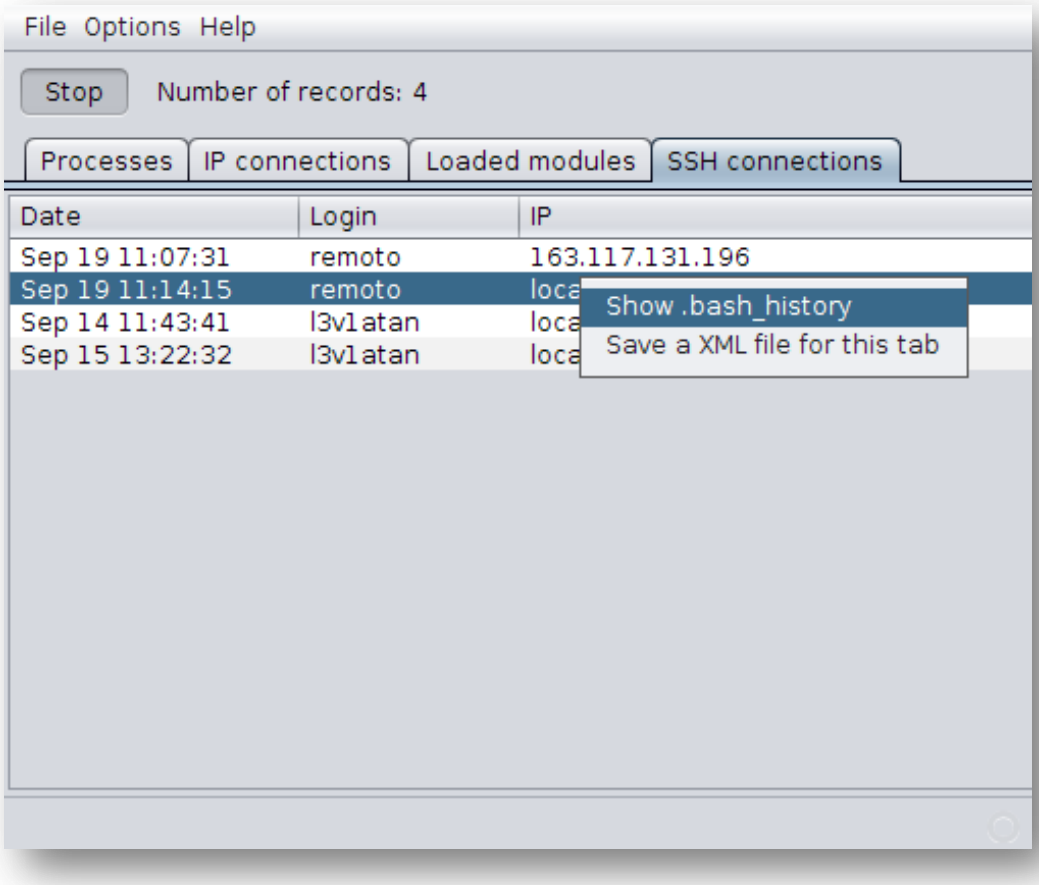


Ilustración 30: IU Popup menu módulo SSH

Si se selecciona la opción “Show .bash_history” aparecerá una ventana nueva en la que se mostrará el fichero de log de comandos .bash_history para ese usuario. En la ilustración anterior se puede apreciar cómo se selecciona una entrada del usuario “remoto”. En la siguiente se puede contemplar el histórico de comandos de “remoto”.

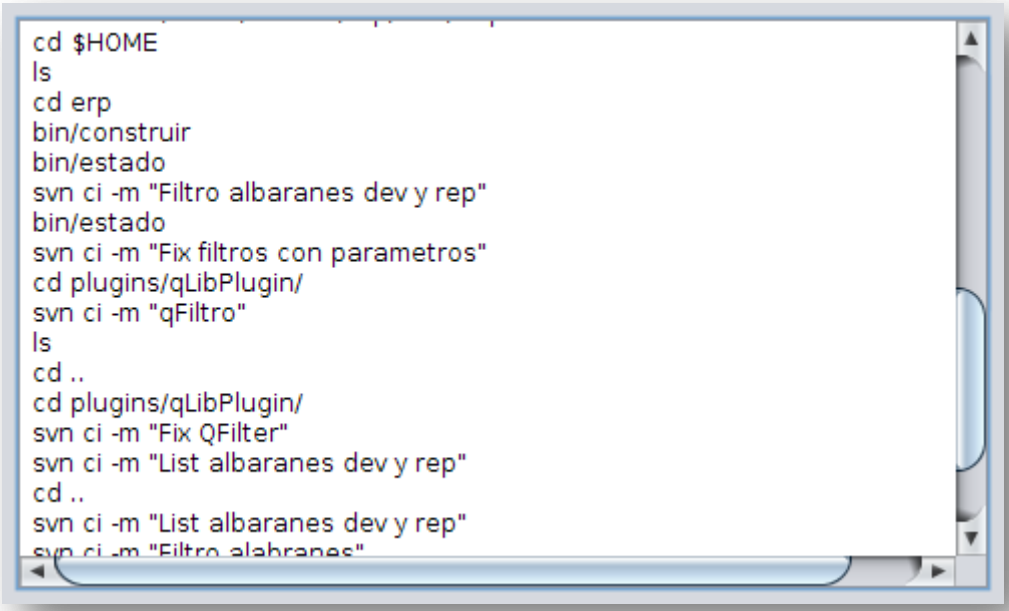


Ilustración 31: IU Bash history

El .bash_history es un fichero oculto de registro que se encuentra en el home de cada usuario y recopila un histórico de los comandos que ha introducido en el terminal. De esta forma podemos ver qué ha hecho el usuario en la máquina. El problema es que el propietario de ese fichero es el propio usuario por lo que tiene permisos para modificar su contenido.

6. Implementación

El presente capítulo tiene por objetivo explicar detalles de la fase de implementación que se han considerado interesantes por su nivel de desafío. Los problemas afrontados y el porqué de las decisiones tomadas son alguno de los puntos a tratar en esta sección.

6.1. Normas de estilo

Para el nombrado de los métodos se ha utilizado la notación lowerCamelCase porque es el estándar de Java. CamelCase es un estilo de escritura que se aplica a palabras o frases compuestas. Existen dos tipos de CamelCase:

- UpperCamelCase: la primera letra de cada una de las palabras es en mayúscula. Ejemplo: EstoEsUpperCamelCase.
- lowerCamelCase: igual que la anterior pero la primera letra es en minúscula. Ejemplo: estoEsLowerCamelCase.

Dentro del patrón Modelo-Vista-Controlador para el nombrado de los métodos del controlador (Warroning2Controller.java) se ha usado un nombrado especial para que en la fase de implementación fuera más sencillo encaminar las funciones desde la vista a la clase del modelo correspondiente. Además de lowerCamelCase se ha usado el siguiente formato: <claseDelModelo><FunciónDelModelo>. Por ejemplo, la función del controlador *procRefrescarDatos()* llama a la función *refrescarDatos()* de la clase Proc.java del modelo.

6.2. Estructuras de datos

Las estructuras de datos usadas para guardar la información que se lee de los ficheros del sistema han sido ArrayList[19]. Desde el principio de la implementación se tuvo claro el uso de estructuras dinámicas. Al comienzo de la implementación se utilizó Vector[20] pero se sustituyó por ArrayList porque era una clase obsoleta (deprecated) y desaconsejaban su uso. La principal diferencia entre estas dos clases es que Vector utiliza la primitiva *synchronized* para cada operación individual. Significa que para cada operación individual sobre los datos bloquea el acceso concurrente sobre el listado, aunque no se necesite, por lo que ralentiza las operaciones. Lo más normal es sincronizar una secuencia de operaciones que puedan producir problemas de concurrencia. Como desde un principio se pensó en la posibilidad de usar hilos, que finalmente se implementó, se optó por la eficiencia de ArrayList y hacer estas sincronizaciones de forma manual.

[19] *API Java 7: ArrayList*.
[20] *API Java 7: Vector*.

6.3. Conectar conexión de red con su proceso (de JDK 6 a 7)

Uno de los mayores retos de este proyecto ha sido la cantidad de investigación que ha tenido detrás de su implementación. El caso de la navegación de una conexión de red hasta el proceso que lo está usando en la máquina es interesante de explicar.

Para esta tarea se optó por el uso de ingeniería inversa. En Linux existe un comando llamado lsof (comentado en el apartado 2.1 Herramientas de monitorización) que lista los ficheros abiertos. El mandato "lsof –i :80" muestra información sobre los sockets que están usando el puerto 80 (en esa información se incluye el nombre del proceso).

```
$ lsof –i :80

COMMAND  PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dropbox 3057 jmlotero   26u  IPv4 719242      0t0  TCP
sissomah:35329->sjc-not8.sjc.dropbox.com:www (ESTABLISHED)
```

Ilustración 32: Ejemplo lsof -i :80

Después se utilizó el comando strace[21] sobre el mandato anterior: "strace lsof –i :80". Gracias a strace se pudo analizar las llamadas NFS que hacía lsof para encontrar desde las conexiones de red su proceso correspondiente.

A continuación se muestra un fragmento del comando "strace lsof –i :80". El motivo de no mostrar la salida entera es porque es demasiado extensa. En la ilustración se puede observar cómo el programa lsof accede a todos los descriptores de fichero de cada proceso, leyendo sus enlaces simbólicos.

[21] strace es una utilidad de línea de comandos para comprobación de errores en el sistema operativo GNU/Linux. Permite monitorizar las llamadas al sistema usadas por un determinado programa y todas las señales que éste recibe.

```
      $ strace lsof -i :80

      ...

      open("/proc",
O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
      fcntl64(3, F_GETFD)                       = 0x1 (flags
FD_CLOEXEC)
      getdents64(3, /* 238 entries */, 32768) = 6080
      stat64("/proc/1/", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
      open("/proc/1/stat", O_RDONLY|O_LARGEFILE) = 4
      read(4, "1 (init) S 0 1 1 0 -1 4202752 52"..., 4096) = 196
      close(4)                                  = 0
      open("/proc/1/fd",
O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 4
      getdents64(4, /* 13 entries */, 32768)  = 312
      readlink("/proc/1/fd/0", "/dev/null"..., 4096) = 9
      lstat64("/proc/1/fd/0", {st_mode=S_IFLNK|0700, st_size=64,
...}) = 0
      stat64("/proc/1/fd/0", {st_mode=S_IFCHR|0666,
st_rdev=makedev(1, 3), ...}) = 0
      readlink("/proc/1/fd/1", "/dev/null", 4096) = 9
      lstat64("/proc/1/fd/1", {st_mode=S_IFLNK|0700, st_size=64,
...}) = 0
      stat64("/proc/1/fd/1", {st_mode=S_IFCHR|0666,
st_rdev=makedev(1, 3), ...}) = 0
      readlink("/proc/1/fd/2", "/dev/null", 4096) = 9
      lstat64("/proc/1/fd/2", {st_mode=S_IFLNK|0700, st_size=64,
...}) = 0
      stat64("/proc/1/fd/2", {st_mode=S_IFCHR|0666,
st_rdev=makedev(1, 3), ...}) = 0
      readlink("/proc/1/fd/3", "pipe:[5889]"..., 4096) = 11
      lstat64("/proc/1/fd/3", {st_mode=S_IFLNK|0500, st_size=64,
...}) = 0
      stat64("/proc/1/fd/3", {st_mode=S_IFIFO|0600, st_size=0, ...})
= 0
      readlink("/proc/1/fd/4", "pipe:[5889]", 4096) = 11
      lstat64("/proc/1/fd/4", {st_mode=S_IFLNK|0300, st_size=64,
...}) = 0
      stat64("/proc/1/fd/4", {st_mode=S_IFIFO|0600, st_size=0, ...})
= 0
      readlink("/proc/1/fd/5", "anon_inode:inotify", 4096) = 18
      lstat64("/proc/1/fd/5", {st_mode=S_IFLNK|0500, st_size=64,
...}) = 0
      stat64("/proc/1/fd/5", {st_mode=0600, st_size=0, ...}) = 0
      readlink("/proc/1/fd/6", "anon_inode:inotify", 4096) = 18
      lstat64("/proc/1/fd/6", {st_mode=S_IFLNK|0500, st_size=64,
...}) = 0
      stat64("/proc/1/fd/6", {st_mode=0600, st_size=0, ...}) = 0
      readlink("/proc/1/fd/7", "socket:[5890]"..., 4096) = 13
      lstat64("/proc/1/fd/7", {st_mode=S_IFLNK|0700, st_size=64,
...}) = 0
      stat64("/proc/1/fd/7", {st_mode=S_IFSOCK|0777, st_size=0,
...}) = 0
      fstat64(5, {st_mode=S_IFREG|0400, st_size=0, ...}) = 0

      ...
```

Ilustración 33: Fragmento de strace lsof -i :80

De la información obtenida de strace se pudo desarrollar el siguiente algoritmo:

1. Para una conexión de red dada obtengo su número de inodo[22]. Por ejemplo 6242.
2. Recorro los descriptores de fichero de cada proceso almacenados en el directorio /proc/[PID]/fd/. Miro a dónde apunta cada enlace simbólico, buscando los que son sockets al inodo (enlace simbólico apunta a "socket:[6242]").
3. Una vez encontrado tengo el PID en la ruta.

Dado este algoritmo se pasó a su implementación en Java.

Otro nuevo reto llegó cuando se intentaba leer el destino de un enlace simbólico. Usando el método getCanonicalPath() de la clase File se conseguía resolver el destino cuando era un fichero o un directorio, pero no cuando era un socket.

Entre los tutoriales oficiales de Java (*The Java Tutorial*) se encuentra uno llamado *Links, Symbolic or Otherwise* que habla sobre métodos para detectar enlaces simbólicos y encontrar su destino. Para detectar enlaces simbólicos usa una función llamada *isSymbolicLink* y para obtener el destino *readSymbolicLink* de la clase Files[23], incluida desde la versión 1.7 de Java.

La implementación de este proyecto se comenzó con JDK 6[24] (versión de Java 1.6) por lo que esa clase no existía. Se tuvo que migrar el proyecto de Netbeans del JDK 6 al JDK 7 (todavía no lanzado de manera oficial por Oracle en ese momento) para poder utilizar esos métodos que consiguieron resolver correctamente los enlaces simbólicos a sockets.

[22] Un inodo, nodo-i o nodo índice es una estructura de datos propia de los sistemas de archivos tipo UNIX como es el caso de Linux. Contiene las características de cualquier objeto que pueda contener el sistema de ficheros (archivos, directorios, enlaces simbólicos, etc), pero no su nombre.
[23] *API Java 7: Files*
[24] Java Development Kit (JDK) es un software que provee herramientas de desarrollo para la creación de programas en Java.

6.4. Refrescar JTables

Durante la implementación del proyecto, la forma de actualizar los datos de las JTables[25] ha sufrido diversas modificaciones, siempre con el fin de buscar un resultado eficiente para el procesador y atractivo para el usuario.

Desde el comienzo se pensó que lo ideal sería que un thread[26] manejara la obtención de datos del sistema y actualizara la tabla de forma dinámica. En una primera fase de la implementación se intentó pero a la hora de que el thread actualizara los datos del JTable se producía una excepción que impedía seguir con la correcta ejecución del programa.

Se optó por la política "divide y vencerás", consistente en acometer un problema complejo creando varios subproblemas de menor complejidad. Por un lado, refrescar los datos de la tabla, y por otro, controlar esa acción con un thread.

6.4.1. Refrescar los datos

Siguiendo el paradigma MVC del framework Swing de Java, una JTable[27] gestiona la presentación de los datos, que tiene cargados a través de otra clase que implementa la interfaz TableModel[28], proveyendo al usuario de funcionalidades como un editor para el contenido de las celdas y ordenaciones por columna. De esta forma se consigue separar la vista de los datos que se encuentran en el modelo.

Se creó una primera versión funcional con un botón "Refresh" que actualizaba los datos mostrados bajo demanda, mediante la función *void refrescarJTable(String tabTitle)*, utilizando como modelo interno del JTable la clase DefaultTableModel[29], que implementa TableModel usando la clase Vector para almacenar los objetos que dan valor a las celdas. También se crearon eventos para capturar la tecla F5 desde cualquier parte de la aplicación y realizar la misma acción.

Como se verá en el apartado, 6.4.2 Optimización de JTables y threads, el uso de la clase DefaultTableModel no es la mejor implementación de la interfaz TableModel.

[25] JTable es la clase de Java Swing que se encarga de mostrar y editar tablas de celdas de dos dimensiones.
[26] Un thread o hilo es una característica del sistema operativo que permite a una aplicación realizar varias tareas a la vez (de forma concurrente).
[27] *API Java 7: JTable*
[28] *API Java 7: TableModel*
[29] *API Java 7: DefaultTableModel*

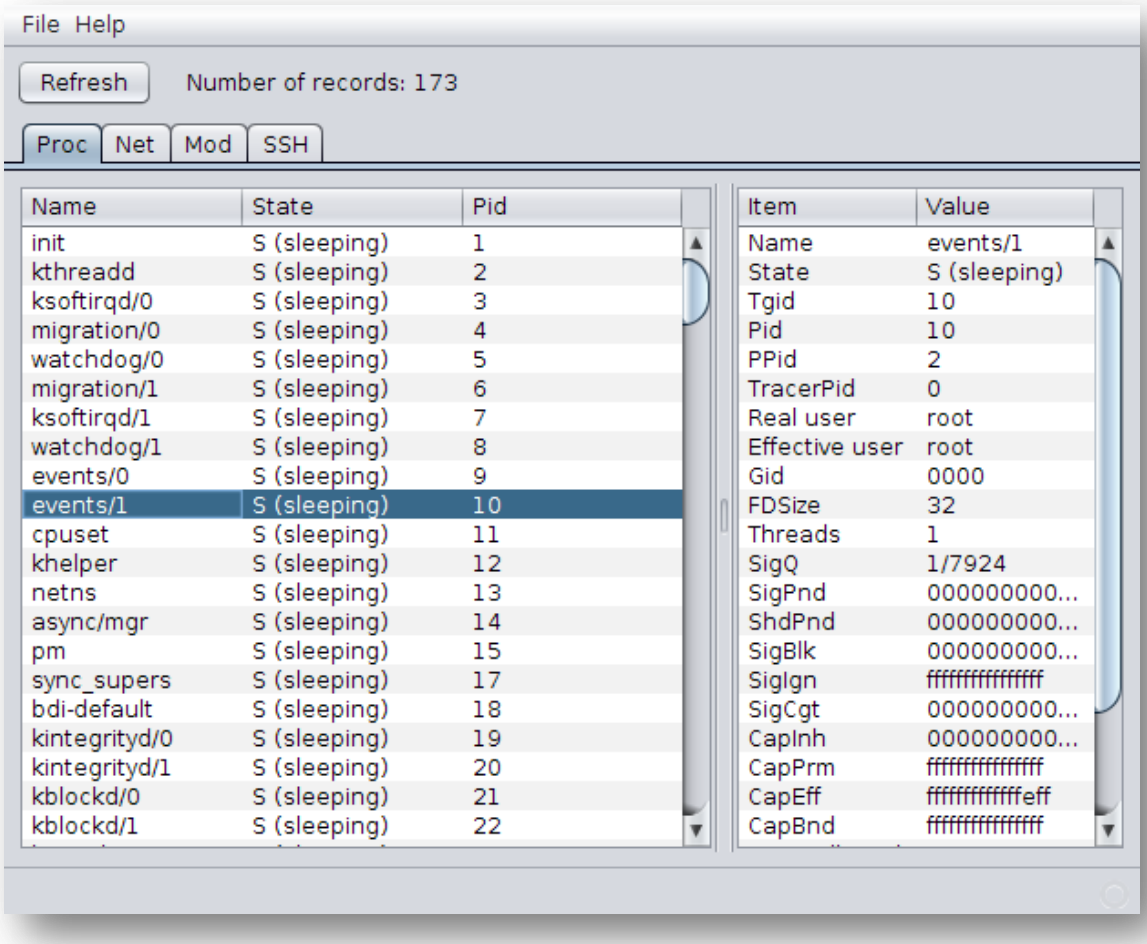


Ilustración 34: Botón "Refresh"

Las primeras versiones de la aplicación con este botón no lograban actualizar bien la tabla porque cuando el número de entradas aumentaba y decrecía lanzaba ArrayIndexOutBoundsException y NullPointerException debido a que la longitud de la JTable no variaba al compás de los datos del modelo. Consultando la web de programación StackOverflow se encontró una entrada llamada *Adding Columns to JTable dynamically* en la que un usuario reproducía una excepción muy similar. La solución por la que se optó fue que en cada actualización se recorría el DefaultTableModel una vez para vaciarlo con la función removeRow(int row) y una segunda vez para añadir los elementos nuevos con addRow(Object columnName).

Por otro lado, cada vez que se recargaba la tabla, se perdía el estado en el que se encontraba. Por ejemplo si había una celda seleccionada, si los datos se estaban mostrando ordenados por una columna o si se había modificado su separación. Fue necesaria la construcción de dos funciones en la capa vista para salvar y restaurar el estado de la JTable:

- Object[] saveTableState (JTable jTable)
- void loadTableState(JTable jTable, Object[] tableStatus)

Elarray de objetos que manejan estas funciones guarda la información necesaria para mantener el estado: TableProperty, TableState, RowSorter, SortKeys y un entero que indica la fila seleccionada.

6.4.2. Optimización de JTables y threads

La idea de un botón “Refresh” fue una solución temporal. Desde el comienzo era necesaria la funcionalidad de una actualización automática, pero por problemas con la implementación de los threads sobre los datos de las JTables se pospuso.

El primer paso fue sustituir el JButton “Refresh” por un JToggleButton “Start/Stop”. Un JToggleButton es un botón que almacena dos estados: pulsado o no pulsado. Mientras esté pulsado el thread debe estar en un bucle de ejecución actualizando los datos de la JTable mostrada. El método startThread() se encarga lanzar el thread y terminarlo si el JToggleButton deja de estar pulsado.

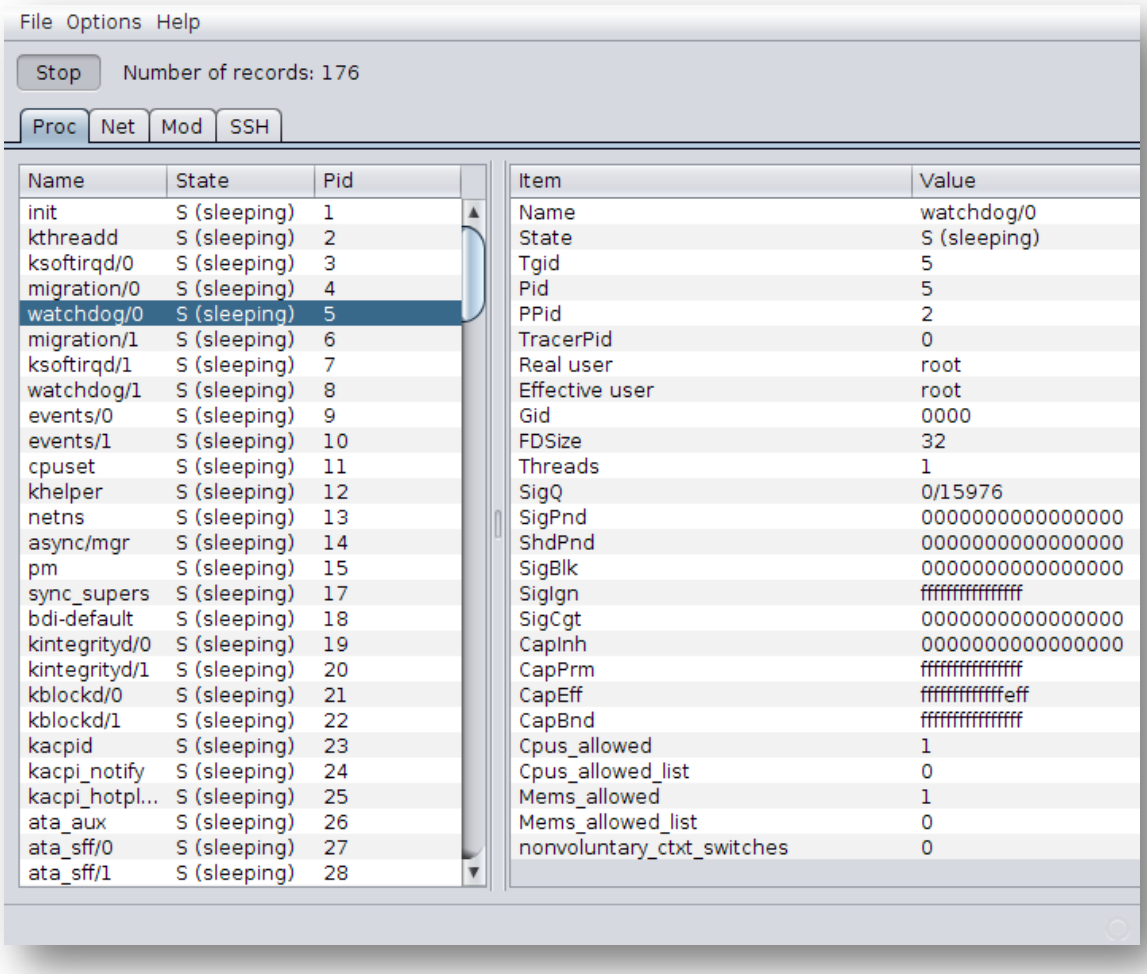


Ilustración 35: Botón "Start-Stop"

Un nuevo problema surgió al llamar al método encargado de refrescar las JTables, *void refrescarJTable(String tabTitle)*. La solución del apartado anterior no era la más óptima debido a que para cada actualización se recorría el modelo dos veces, una para vaciarlo y otra para agregar las columnas. Además, esta función al ser ejecutada desde un thread provocaba una excepción debido a problemas de concurrencia entre la JTable y DefaultTableModel. En la web StackOverflow hay una entrada que reproduce la misma excepción llamada *Refreshing JTable in Swing Gives Exception*. En las respuestas que recibe se recomienda realizar una implementación de TableModel que sustituya a DefaultTableModel.

Existe un artículo de Francesc Rosés llamado *JTable, TableModel Y Rendimiento* en el habla de la optimización en la vista de JTables de cara al consumo de memoria, fluidez en el scroll y conexiones a una base de datos a través de un driver JDBC. La parte que se ha usado del artículo es la que se centra sobre el mal uso de la clase DefaultTableModel.

DefaultTableModel extiende de AbstractTableModel, que peca en el uso exhaustivo de la clase Vector, que como se explicó en el apartado 6.2 Estructuras de datos, tiene un coste computacional elevado ya que tiene todos sus métodos sincronizados.

La solución fue crear una implementación propia de TableModel, a la que se llamó MiTableModel, la cual se adaptaba mejor a las necesidades de este proyecto. En vez de usar la clase Vector se utilizan arrays de tipo String. El motivo de usar una estructura de datos estática es porque cada vez que se realiza una actualización el objeto se reemplaza, no tiene problemas de concurrencia y no necesita conversiones de ningún tipo porque es el mismo tipo de datos que maneja JTable. Por último destacar que MiTableModel incluye el método *void actualizar(String [][] contenido)* que se encarga de realizar una correcta actualización de los datos avisando a todos sus subscritores de que la tabla ha cambiado para evitar excepciones.

6.5. Obtener nombre de usuario a apartir del UID

El kernel maneja identificadores de usuario (UIDs) para asignarlos a procesos y establecer permisos en el sistema de archivos, entre otros.

El UID no es más que un número interno del sistema. El usuario necesita una notación más familiar por lo que surge la idea del nombre de usuario o login. Existe una correspondencia 1:1 entre ambas entidades.

En el fichero /etc/passwd se almacena la información necesaria para esta relación. A continuación se muestra un fragmento de ejemplo del contenido:

```
$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
...
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/bin/sh
...
hplip:x:112:7:HPLIP system user,,,:/var/run/hplip:/bin/false
gdm:x:113:120:Gnome Display Manager:/var/lib/gdm:/bin/false
jmlotero:x:1000:1000:José María
López,,,:/home/jmlotero:/bin/bash
jetty:x:114:123::/usr/share/jetty:/bin/false
sshd:x:115:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:116:125:MySQL Server,,,:/nonexistent:/bin/false
```

Ilustración 36: Fragmento de /etc/passwd

En este listado se encuentran todos los usuarios que pueden tener acceso al sistema.

La siguiente ilustración muestra el formato que sigue cada registro:

Ilustración 37: Formato de registro de /etc/passwd

En la aplicación que se ha desarrollado, se contaba en muchas ocasiones con el UID, pero al usuario lo que le interesaba era el login. El método *String uid2Username(String uid)* se encarga de realizar esta conversión. Básicamente tokeniza cada línea por el delimitador ":" y analiza cada token.

En las JTable de detalles de procesos y conexiones de red se resuelve el nombre de usuario para que sea más amigable. También es necesario consultar este fichero en el módulo SSH, para encontrar la carpeta personal (home) del usuario y leer su fichero .bash_history .

6.6. Generación de XML

En la fase final de implementación del proyecto se pensó que sería muy útil la posibilidad de exportar a fichero la información que estaba mostrando la aplicación. Por ejemplo, si necesitas enviar los datos que se están mostrando a otra persona por correo electrónico.

Se estudiaron diferentes formatos para el fichero y finalmente se optó por el estándar XML[30] debido a su sencillez de compresión humana y facilidad de creación desde la herramienta. Además, como líneas futuras, XML combinado con XSLT[31] aporta facilidades para la elaboración de históricos web.

Como se explicó en el apartado 5.2 Diseño de la interfaz de usuario, en el software desarrollado se puede generar un fichero XML de un único módulo o para todos.

Por motivos de espacio, puede consultar en el anexo V el contenido de un XML generado por la aplicación para un módulo.

Todos los ficheros XML creados desde la herramienta siguen la estructura del siguiente DTD[32]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT row ((column+))>
<!ATTLIST row
      num CDATA #REQUIRED
>
<!ELEMENT modules ((module+))>
<!ELEMENT module ((row+))>
<!ATTLIST module
      name CDATA #REQUIRED
>
<!ELEMENT column (#PCDATA)>
<!ATTLIST column
      name CDATA #REQUIRED
>
```

Ilustración 38: DTD para el XML

[30] *Extensible Markup Language*
[31] *Extensible Stylesheet Language Transformations*
[32] *Document Type Definition*

6.7. Convirtiendo la herramienta en portable

Uno de los principales requisitos de la herramienta era que la portabilidad, por lo que el software debía ser perfectamente ejecutable desde un dispositivo extraíble sin necesidad de una instalación previa. De forma intrínseca, la aplicación también debería correr en dispositivos de sólo lectura y ser autocontenida. Se probaron distintos soportes portables, reales (pendrive, CD, DVD) como emulados (imagen ISO). A continuación se exponen los problemas que surgieron para esta fase.

6.7.1. Imágenes ISO

Durante el desarrollo de la aplicación se crearon imágenes ISO[33] para poder estudiar el comportamiento de la herramienta en un dispositivo de sólo lectura.

La forma más sencilla de crear una imagen ISO es con el comando "mkisofs –o <pathDeSalidaFicheroISO> <pathDeDirectorioDeEntrada>".

Montar un fichero ISO se puede hacer de la siguiente manera:

1. Definir directorio que hará de punto de montaje: "sudo mkdir <rutaPuntoDeMonaje>".
2. Cargar el módulo loop: "sudo modprobe loop".
3. Montar la ISO en el punto de montaje definido: "sudo mount <rutaDeFicheroISO> <rutaPuntoDeMontaje> -t iso9660 –o loop"

Para desmontar la imagen basta con ejecutar el comando "sudo umount <rutaPuntoDeMontaje>".

Las pruebas realizadas sobre este soporte fueron satisfactorias. Se pueden consultar en el apartado 7.2 Pruebas de sistema.

6.7.2. Pendrives

En la primera versión que se portó el software a un pendrive estaba formateado con un sistema de archivos FAT32[34]. El problema que surgió era que no se podía ejecutar la aplicación desde ahí ya que FAT32 no dispone de permisos de usuario. Se tuvo que realizar un

[33] Archivo donde se almacena una copia o imagen exacta de un sistema de ficheros, normalmente un dispositivo óptico. Se rige por el estándar ISO 9660.
[34] *Tabla de asignación de archivos (File allocation table)*

cambio del sistema de archivos a ext3[35], el formato más extendido en Linux. Aunque actualmente está siendo reemplazado por su sucesor ext4[36].

Existen en el mercado una gama de pendrives con la opción de trabajar en modo sólo lectura. Esta opción es muy útil porque sirve para proteger el dispositivo frente a malware de máquinas asaltadas impidiendo que se escriba nada. El principal problema reside en que pendrives con estas características resultan difíciles de adquirir. Se ha encontrado una web que hace referencia a memorias USB con protección de escritura de hardware. Para el proyecto se contó con el modelo Imation Pivot, ahora descatalogado.

6.7.3. CD/DVD

Se han probado la ejecución desde CD con el sistema de archivos de sólo lectura UDF[37]. En las primeras pruebas el sistema operativo se montó automáticamente como read-only y sin permisos de ejecución. Al ser un sistema de sólo lectura no fue posible hacer chmod[38] para modificar los permisos porque implicaba escribir en el soporte.

Se han encontrado tres soluciones a este problema:

1. Montar la unidad a mano mediante “mount”.
2. Modificar el fichero /etc/fstab y añadir una línea del tipo “dev/cdrom /media/cdrom auto ro,noauto,user,exec 0 0”.
3. Copiar los ficheros del dispositivo a disco para poder ejecutar chmod sobre ellos.

En otras pruebas hechas parece que dependiendo del software de grabación afecta a los privilegios en el automontaje en Linux. Se pueden consultar en el apartado 7.2 Pruebas de sistema.

6.7.4. Script de ejecución

Para facilitar el arranque de la aplicación se incluye el script run.sh que lanza la aplicación como sudo con el propio JRE que porta el proyecto. Puede consultar el contenido del fichero en el anexo VI. Script run.sh.

[35] *Tercer sistema de archivos extendido (third extended filesystem)*
[36] *Cuarto sistema de archivos extendido (fourth extende filesystem)*
[37] Universal Disk Format es un sistema de archivos que utiliza el estándar ISO 9660 propiedad de Adaptec usado por grabadoras de CD y DVD.
[38] Del inglés “change mode” es un comando que realiza una llamada al sistema que permite cambiar los permisos de acceso de un archivo o directorio.

7. Plan de pruebas

En el presente capítulo se especifica el plan de pruebas elaborado con el objetivo de verificar que el producto final obtenido se adapta a las necesidades descritas en el apartado 4. Análisis, correspondiente a la fase de análisis.

En la metodología de desarrollo Métrica V3 [6] se define un plan de pruebas a desarrollar a distintos niveles:

- Pruebas unitarias.
- Pruebas de integración.
- Pruebas de sistema.
- Pruebas de implementación.
- Pruebas de aceptación o usabilidad.

El plan de pruebas diseñado para este proyecto incluirá la sección de pruebas unitarias y de sistema.

7.1. Pruebas unitarias

En esta sección, las pruebas a realizar serán del tipo de caja negra o enfoque funcional, es decir, se estudiarán desde el punto de vista de las entradas que recibe y las salidas que produce, sin tener en cuenta el comportamiento interno. Las pruebas unitarias se realizarán a nivel de función por lo que se va a incluir un campo correspondiente al nombre de la función en el código.

La especificación de las pruebas unitarias seguirá el siguiente formato tabular:

- **Identificador:** código que identifica de manera unívoca cada tabla, PU-XX. XX denota un valor numérico de dos cifras que empieza en 01 y se irá incrementando a modo de contador.
- **Función:** nombre de la función del código que se está comprobando.
- **Propósito:** finalidad u objetivo que persigue.
- **Salida esperada:** salida esperada para la función.

Todas las pruebas descritas a continuación han sido realizadas en una Ubuntu 10.10 de 32 bits.

a) Vista

Identificador	PU-01
Función	jTable1MouseClicked
Propósito	Evento que controla el click del ratón con el botón izquierdo sobre una entrada de la JTable 1 (Proc). Obtiene el PID del proceso seleccionado para dibujar el detalle de ese proceso. Si es el botón derecho muestra las opciones correspondientes en un JPopupMenu
Salida esperada	Si es el botón izquierdo, carga el detalle del proceso en la JTable 4. Si es el botón derecho muestra un JPopupMenu con las opciones

Tabla 44: PU-01 jTable1MouseClicked

Identificador	PU-02
Función	jTable2MouseClicked
Propósito	Evento que controla el click del ratón con el botón izquierdo sobre una entrada de la JTable 2 (Net). Obtiene el "sl" del socket seleccionado para dibujar el detalle de esa conexión. Si es el botón derecho muestra las opciones correspondientes en un JPopupMenu
Salida esperada	Si es el botón izquierdo, carga el detalle del socket en la JTable 5. Si es el botón derecho muestra un JPopupMenu con las opciones

Tabla 45: PU-02 jTable2MouseClicked

Identificador	PU-03
Función	jTable3MouseClicked
Propósito	Evento que controla si se ha pulsado con el botón derecho del ratón en la JTable 3 (SSH) teniendo una fila seleccionada
Salida esperada	Se muestra un JPopupMenu al usuario con las opciones que dispone

Tabla 46: PU-03 jTable3MouseClicked

Identificador	PU-04
Función	jTable6MouseClicked
Propósito	Evento que controla si se ha pulsado con el botón derecho del ratón en la JTable 6 (Mod) teniendo una fila seleccionada
Salida esperada	Se muestra un JPopupMenu al usuario con las opciones que dispone

Tabla 47: PU-04 jTable6MouseClicked

Identificador	PU-05
Función	jTabbedPane1MouseClicked
Propósito	Si se produce un click en el JTabbedPane (por ejemplo un cambio de pestaña) se recarga el contenido de la tabla que se muestra
Salida esperada	La nueva pestaña del JTabbedPane con los datos del listado refrescados

Tabla 48: PU-05 jTabbedPane1MouseClicked

Identificador	PU-06
Función	saveTableState
Propósito	Guarda el estado de una JTable
Salida esperada	Objeto tipo array con cinco elementos que definen el estado de la JTable: TableProperty, TableState, RowSorter, SortKeys, selectedColumn

Tabla 49: PU-06 saveTableState

Identificador	PU-07
Función	loadTableState
Propósito	Restaura el estado de una JTable
Salida esperada	La JTable en el mismo estado que se encontraba cuando se llamó a la función saveTableState

Tabla 50: PU-07 loadTableState

Identificador	PU-08
Función	accionBotonRefresh
Propósito	Refresca los datos de la JTable que se está mostrando al usuario manteniendo el estado de la tabla (ordenación, fila seleccionada, altura de scroll)
Salida esperada	La JTable con los nuevos datos y mismo estado

Tabla 51: PU-08 accionBotonRefresh

Identificador	PU-09
Función	accionShowBashHistory
Propósito	Recoge el login seleccionado en la fila del listado de conexiones SSH de la JTable 3 y mostrar un JFrame con el contenido del fichero .bash_history de ese usuario
Salida esperada	JFrame con que muestra el contenido del .bash_history para ese usuario

Tabla 52: PU-09 accionShowBashHistory

Identificador	PU-10
Función	navigateToProcess
Propósito	Realiza un cambio de vista de una conexión de red seleccionada en la JTable 2 (Net) al proceso correspondiente de la JTable 1 (Proc)
Salida esperada	Cambio de vista al proceso padre del socket

Tabla 53: PU-10 navigateToProcess

Identificador	PU-11
Función	refrescarJTable
Propósito	Recibe el nombre de la pestaña que está activa y se encarga de recargar los datos que del JTable. Actualiza la etiqueta JLabel 1 con el número de entradas que tiene el listado
Salida esperada	Datos actualizados del JTable

Tabla 54: PU-11 refrescarJTable

Identificador	PU-12
Función	accionBotonStartStop
Propósito	Controla el estado del JToggleButton encargado de refrescar o no los datos del listado que se está mostrando al usuario
Salida esperada	Si está activado actualiza los datos de forma periódica. En caso contrario no hace nada

Tabla 55: PU-12 accionBotonStartStop

Identificador	PU-13
Función	startThread
Propósito	Maneja el thread que se encarga actualizar los datos del modelo
Salida esperada	Ejecución correcta del thread

Tabla 56: PU-13 startThread

Identificador	PU-14
Función	procGenerateXML
Propósito	Acción que se ejecuta cuando el usuario elige generar el XML únicamente para la JTable 1 (Proc)
Salida esperada	Diálogo para guardar XML

Tabla 57: PU-14 procGenerateXML

Identificador	PU-15
Función	netGenerateXML
Propósito	Acción que se ejecuta cuando el usuario elige generar el XML únicamente para la JTable 2 (Net)
Salida esperada	Diálogo para guardar XML

Tabla 58: PU-15 netGenerateXML

Identificador	PU-16
Función	sshGenerateXML
Propósito	Acción que se ejecuta cuando el usuario elige generar el XML únicamente para la JTable 3 (SSH)
Salida esperada	Diálogo para guardar XML

Tabla 59: PU-16 sshGenerateXML

Identificador	PU-17
Función	modGenerateXML
Propósito	Acción que se ejecuta cuando el usuario elige generar el XML únicamente para la JTable 6 (Mod)
Salida esperada	Diálogo para guardar XML

Tabla 60: PU-17 modGenerateXML

Identificador	PU-18
Función	generateXML
Propósito	Acción que se ejecuta cuando el usuario elige generar el XML global, para todas las pestañas
Salida esperada	Diálogo para guardar XML

Tabla 61: PU-18 generateXML

Identificador	PU-19
Función	dialogoGuardarXML
Propósito	Muestra una ventana que permite al usuario navegar por el sistema de archivos para seleccionar la ruta del fichero a guardar. Se aplica un filtro a la exploración de ficheros con extensión XML
Salida esperada	Ruta del fichero

Tabla 62: PU-19 dialogoGuardarXML

b) Controlador

Se excluyen las pruebas unitarias en esta capa porque no son más que un encaminamiento de la vista a la parte correspondiente del modelo.

c) Modelo – Proc

Identificador	PU-20
Función	refrescarDatos
Propósito	Actualiza el ArrayList que almacena la información de los procesos
Salida esperada	ArrayList actualizado

Tabla 63: PU-20 refrescarDatos

Identificador	PU-21
Función	obtenerDirectorios
Propósito	Devuelve los directorios contenidos en el directorio pasado por parámetro
Salida esperada	ArrayList con las rutas de los directorios

Tabla 64: PU-21 obtenerDirectorios

Identificador	PU-22
Función	filtrarDirectoriosPID
Propósito	Filtra un ArrayList con directorios cogiendo los directorios que tienen un PID
Salida esperada	ArrayList de directorios referentes a un PID

Tabla 65: PU-22 filtrarDirectoriosPID

Identificador	PU-23
Función	leerStatusProcesos
Propósito	Devuelve la información de los procesos pasados por parámetro. Internamente llama a "leerStatusProceso" para cada elemento del ArrayList
Salida esperada	ArrayList de HashMap para los procesos pasados por parámetro

Tabla 66: PU-23 leerStatusProcesos

Identificador	PU-24
Función	leerStatusProceso
Propósito	Devuelve la información del proceso pasado por parámetro
Salida esperada	HashMap con la información de un proceso

Tabla 67: PU-24 leerStatusProceso

Identificador	PU-25
Función	getInfoProcesos
Propósito	Devuelve el ArrayList que contiene la información de todos los procesos
Salida esperada	ArrayList de HashMaps para cada proceso con la info encontrada

Tabla 68: PU-25 getInfoProcesos

Identificador	PU-26
Función	getArrayDatos
Propósito	Construye un array de dos dimensiones para la JTable que lo muestra
Salida esperada	Los datos representados como un array de dos dimensiones

Tabla 69: PU-26 getArrayDatos

Identificador	PU-27
Función	getClavesInfoProcesos
Propósito	Construye un array con las claves necesarias para el listado
Salida esperada	Array con las claves

Tabla 70: PU-27 getClavesInfoProcesos

Identificador	PU-28
Función	getClavesInfoProcesosTodas
Propósito	Construye un array con las claves en el mismo orden que el fichero status
Salida esperada	Array con todas las claves

Tabla 71: PU-28 getClavesInfoProcesosTodas

Identificador	PU-29
Función	inicilizarHashMap
Propósito	Inicializa un hashmap con sus respectivas claves y todos los valores a "" por si no existen
Salida esperada	Un objeto HashMap con todos sus valores inicializados a ""

Tabla 72: PU-29 inicilizarHashMap

Identificador	PU-30
Función	getDetalleProceso
Propósito	Para un PID pasado por parámetro busca su entrada en el ArrayList de procesos y devuelve toda la información que será representada por la vista
Salida esperada	Un array de parejas de valores con el detalle del proceso

Tabla 73: PU-30 getDetalleProceso

Identificador	PU-31
Función	uid2Username
Propósito	Analiza el fichero /etc/passwd para obtener el nombre del usuario asociado al UID dado
Salida esperada	Nombre de usuario asociado al UID

Tabla 74: PU-31 uid2Username

Identificador	PU-32
Función	getRelProcNet
Propósito	Recibe un PID sacado de un inodo de una conexión de red y busca información del proceso al que pertenece
Salida esperada	Array con información del proceso para pintar en el detalle de la tabla Net

Tabla 75: PU-32 getRelProcNet

Identificador	PU-33
Función	uidRowToArrayUsername
Propósito	Recibe una fila con los UIDS de un proceso y los devuelve en forma de array
Salida esperada	Array con un uid en cada posición

Tabla 76: PU-33 uidRowToArrayUsername

Identificador	PU-34
Función	generateXML
Propósito	Crea un String con código XML de toda la información sobre los procesos en memoria
Salida esperada	String con XML de los procesos en memoria

Tabla 77: PU-34 generateXML

d) Modelo – Net

Identificador	PU-35
Función	refrescarDatos
Propósito	Actualiza el ArrayList que almacena la información de las conexiones de red
Salida esperada	ArrayList actualizado

Tabla 78: PU-35 refrescarDatos

Identificador	PU-36
Función	parsearProcNetTcpUdp
Propósito	Lee líneas de /proc/net/tcp o /proc/net/udp y llama a la función encargadas de parsear las líneas. Devuelve su contenido en forma de ArrayList
Salida esperada	ArrayList con información de las conexiones

Tabla 79: PU-36 parsearProcNetTcpUdp

Identificador	PU-37
Función	parsearLineaProcNetTcpUdp
Propósito	Parsea una línea de /proc/net/tcp o /proc/net/udp y devuelve un HashMap con una pareja de valores
Salida esperada	HashMap con información de una conexión

Tabla 80: PU-37 parsearLineaProcNetTcpUdp

Identificador	PU-38
Función	getClavesProcNetTcp
Propósito	Construye un array con las claves del hashmap de /proc/net/tcp
Salida esperada	Array con las claves

Tabla 81: PU-38 getClavesProcNetTcp

Identificador	PU-39
Función	getClavesProcNetUdp
Propósito	Construye un array con las claves del hashmap de /proc/net/udp
Salida esperada	Array con las claves

Tabla 82: PU-39 getClavesProcNetUdp

Identificador	PU-40
Función	getArrayDatos
Propósito	Construye un array de dos dimensiones para la JTable que lo muestra
Salida esperada	Los datos representados como un array de dos dimensiones

Tabla 83: PU-40 getArrayDatos

Identificador	PU-41
Función	traducirHexAddress
Propósito	Traduce una cadena IP:PUERTO en formato hexadecimal y little-endian a formato decimal y big-endian
Salida esperada	Cadena IP:PUERTO en formato decimal y big-endian

Tabla 84: PU-41 traducirHexAddress

Identificador	PU-42
Función	traducirSt
Propósito	Traduce el número entero correspondiente al estado de una conexión a su correspondiente estado en inglés
Salida esperada	Estado en inglés

Tabla 85: PU-42 traducirSt

Identificador	PU-43
Función	getClavesProcNetTodas
Propósito	Construye un array con las posibles claves del HashMap para TCP y UDP
Salida esperada	Array con todas las claves

Tabla 86: PU-43 getClavesProcNetTodas

Identificador	PU-44
Función	getDetalleConexion
Propósito	Para un identificador de socket pasado por parámetro busca su entrada en el ArrayList de conexiones y devuelve toda la información que será representada por la vista
Salida esperada	Un array de parejas de valores con el detalle del socket

Tabla 87: PU-44 getDetalleConexion

Identificador	PU-45
Función	getPidSocket
Propósito	Devuelve el PID correspondiente a la última conexión seleccionada por el usuario en la vista
Salida esperada	PID correspondiente

Tabla 88: PU-45 getPidSocket

Identificador	PU-46
Función	inodeSocket2Pid
Propósito	Para un inodo de un socket obtiene su PID
Salida esperada	PID asociado al inodo

Tabla 89: PU-46 inodeSocket2Pid

Identificador	PU-47
Función	generateXML
Propósito	Crea un String con código XML de toda la información sobre las conexiones de red en memoria
Salida esperada	XML de las conexiones de red en memoria

Tabla 90: PU-47 generateXML

e) Modelo – Mod

Identificador	PU-48
Función	getClavesProcModules
Propósito	Construye un array con las claves del hashmap de /proc/modules
Salida esperada	Array de claves para el HashMap

Tabla 91: PU-48 getClavesProcModules

Identificador	PU-49
Función	parsearProcModules
Propósito	Lee cada línea de /proc/modules, llama a una función que se encarga de parsear la línea y construye el ArrayList con los datos
Salida esperada	ArrayList con la información de /proc/modules

Tabla 92: PU-49 parsearProcModules

Identificador	PU-50
Función	parsearLineaProcModules
Propósito	Parsea una línea del fichero /proc/modules
Salida esperada	Devuelve un HashMap con la información de la línea

Tabla 93: PU-50 parsearLineaProcModules

Identificador	PU-51
Función	getArrayDatos
Propósito	Construye un array de dos dimensiones para la JTable que lo muestra
Salida esperada	Los datos representados como un array de dos dimensiones

Tabla 94: PU-51 getArrayDatos

Identificador	PU-52
Función	refrescarDatos
Propósito	Actualiza el ArrayList que almacena la información de los módulos cargados en memoria
Salida esperada	ArrayList actualizado

Tabla 95: PU-52 refrescarDatos

Identificador	PU-53
Función	generateXML
Propósito	Crea un String con código XML de toda la información sobre los módulos cargados en memoria
Salida esperada	String con el XML de los módulos cargados en memoria

Tabla 96: PU-53 generateXML

f) Modelo – SSH

Identificador	PU-54
Función	parsearVarLogAuth
Propósito	Lee cada línea de /var/log/auth.log, llama a una función que se encarga de parsear la línea y construye el ArrayList con los datos
Salida esperada	ArrayList<HashMap> con la información de /var/log/auth.log

Tabla 97: PU-54 parsearVarLogAuth

Identificador	PU-55
Función	parsearLineaLogSsh
Propósito	Parsea una línea del fichero /var/log/auth.log
Salida esperada	Devuelve un HashMap con la información de la línea

Tabla 98: PU-55 parsearLineaLogSsh

Identificador	PU-56
Función	contieneCadena
Propósito	Recibe una línea, un patrón a buscar y un boolean que marca si se ignorarán las mayúsculas. Busca en la cadena "patronBusqueda" en el String "linea"
Salida esperada	Devuelve true si la línea contiene el patrón de búsqueda

Tabla 99: PU-56 contieneCadena

Identificador	PU-57
Función	myGrep
Propósito	Busca en un fichero todas las líneas que contengan el patrón de búsqueda
Salida esperada	Las líneas del fichero que contienen el patrón de búsqueda

Tabla 100: PU-57 myGrep

Identificador	PU-58
Función	getClavesSshLog
Propósito	Construye un array con las claves del hashmap de /var/log/auth.log
Salida esperada	Array de claves para el HashMap

Tabla 101: PU-58 getClavesSshLog

Identificador	PU-59
Función	getArrayDatos
Propósito	Construye un array de dos dimensiones para la JTable que lo muestra
Salida esperada	Los datos representados como un array de dos dimensiones

Tabla 102: PU-59 getArrayDatos

Identificador	PU-60
Función	leerBashHistory
Propósito	Lee el fichero .bash_history del home de un usuario para un login pasado por parámetro
Salida esperada	Un string con el contenido del fichero .bash_history

Tabla 103: PU-60 leerBashHistory

Identificador	PU-61
Función	refrescarDatos
Propósito	Actualiza el ArrayList que almacena la información del fichero /var/log/auth.log
Salida esperada	ArrayList actualizado

Tabla 104: PU-61 refrescarDatos

Identificador	PU-62
Función	generateXML
Propósito	Crea un String con código XML de toda la información sobre las conexiones SSH establecidas
Salida esperada	String con el XML de las conexiones SSH establecidas

Tabla 105: PU-62 generateXML

7.2. Pruebas de sistema

Las pruebas de sistema tienen por objetivo realizar una serie de comprobaciones rutinarias con la herramienta desarrollada en una máquina real con el fin de constatar el correcto funcionamiento de la aplicación.

Las pruebas de sistema se han realizado sobre distribuciones basadas en Debian y Red Hat porque en la actualidad son las más extendidas[39].

Las tareas a realizar en cada prueba son las siguientes:

1. Arrancar la aplicación
2. Pestaña de procesos
    a. Comprobar el listado de procesos
    b. Comprobar el detalle de un proceso
    c. Generar XML para el listado de procesos
3. Pestaña de conexiones de red
    a. Comprobar el listado de conexiones
    b. Comprobar el detalle de una conexión
    c. Generar XML para el listado de conexiones
    d. Navegar de una conexión de red a su proceso
4. Pestaña de módulos cargados en el kernel
    a. Comprobar el listado de módulos
    b. Generar el XML para el listado de módulos
5. Pestaña de conexiones SSH
    a. Comprobar el listado de conexiones SSH
    b. Generar el XML para el listado de conexiones SSH
    c. Mostrar el listado .bash_history de un usuario
6. Comprobar el correcto funcionamiento del botón Start/Stop
7. Generar el XML para todas las pestañas
8. Cerrar aplicación

La especificación de todas las pruebas seguirá el siguiente formato tabular:

[39] En la página *DistroWatch.com* se puede encontrar noticias, ranking de popularidad y otro tipo de información general sobre diversas distribuciones de Linux. Según esta web Ubuntu (Debian), Fedora (Red Hat), Mint (Ubuntu) y Debian forman parte del top 5 de distribuciones más usadas en 2009 y 2010.

- **Identificador:** código que identifica de manera unívoca cada tabla, PS-XX. XX denota un valor numérico de dos cifras que empieza en 01 y se irá incrementando a modo de contador.
- **Sistema operativo:** distribución y versión del sistema operativo en el que se ejecuta.
- **Entorno software:** especificaciones del entorno software donde se realiza la prueba.
- **Anotaciones:** notas, conclusiones o detalles obtenidos durante el proceso.

Identificador	PS-01
Sistema operativo	Ubuntu 11.04 32 bits, Gnome
Entorno software	Live en máquina virtual. Aplicación ejecutada desde CD
Anotaciones	La unidad se monta a mano con privilegios de lectura y ejecución

Tabla 106: PS-01 Ubuntu 11.04 32 bits

Identificador	PS-02
Sistema operativo	Ubuntu 10.10 32 bits, Gnome
Entorno software	Instalación en disco. Aplicación ejecutada desde CD
Anotaciones	La unidad se monta automáticamente con privilegios de lectura y ejecución. Se puede ejecutar el script desde Nautilus con la opción “Ejecutar en un terminal”

Tabla 107: PS-02 Ubuntu 10.10 32 bits

Identificador	PS-03
Sistema operativo	BackTrack 5 32 bits, Gnome
Entorno software	Live en CD. Aplicación ejecutada desde pendrive
Anotaciones	Se usa xHydra para atacar por SSH a la máquina y la pestaña de red muestra las peticiones al puerto local 22

Tabla 108: PS-03 BackTrack 5 32 bits

Identificador	PS-04
Sistema operativo	BackTrack 5 32 bits, Gnome
Entorno software	Live en CD. Aplicación ejecutada desde pendrive
Anotaciones	Se carga un módulo (modprobe ath9k) y en la pestaña de módulos se observa cómo aparece el módulo y sus dependencias

Tabla 109: PS-04 BackTrack 5 32 bits

Identificador	PS-05
Sistema operativo	BackTrack 5 32 bits, Gnome
Entorno software	Live en CD. Aplicación ejecutada desde pendrive
Anotaciones	Se para el servicio SSHD y desparece del listado de procesos en ejecución

Tabla 110: PS-05 BackTrack 5 32 bits

Se ha probado BackTrack por ser una distribución de Linux especialmente pensada y diseñada para la auditoria de seguridad y relacionada con la seguridad informática en general. BackTrack incluye una extensa colección de herramientas completamente usables desde su versión Live por lo que no requiere ninguna instalación para poder utilizarse. Esta filosofía de portabilidad encaja perfectamente con la seguida en este proyecto por lo que se ha pensado la posibilidad de contactar con los desarrolladores de BackTrack para que incluyan la herramienta desarrollada en futuras versiones.

Identificador	PS-06
Sistema operativo	Fedora 15 32 bits, Gnome
Entorno software	Live en máquina virtual. Aplicación ejecutada desde CD
Anotaciones	No funciona la opción “Ejecutar en un terminal” de la prueba anterior porque en esta versión live no permite usar el comando sudo al usuario por defecto. Abriendo el terminal y cambiando el usuario a root se ejecuta correctamente

Tabla 111: PS-06 Fedora 15 32 bits

Identificador	PS-06
Sistema operativo	Fedora 15 32 bits, Gnome
Entorno software	Live en máquina virtual. Aplicación ejecutada desde CD
Anotaciones	En el listado de conexiones SSH no aparecen registros pese a existir. Fedora, basada en Red Hat, almacena los log de acceso en /var/secure/log. El formato de línea es el mismo que en Debian. Se corrige el fallo

Tabla 112: PS-07 Fedora 15 32 bits

8. Conclusiones y líneas futuras

8.1. Conclusiones

Mediante el presente proyecto se ha querido manifestar el vacío existente en Linux respecto a una herramienta que integra diversas funcionalidades de análisis del sistema y que presenta datos en tiempo real de forma gráfica y sencilla al usuario. Gracias a las especificaciones proporcionadas ha sido posible desarrollar una aplicación que consigue simplificar las tareas rutinarias que se realizan cuando se hace una primera toma de contacto con una máquina asaltada para que cualquier persona pueda usarla, sin ser necesariamente experta en el campo de seguridad. Todo ello sin ejecutar aplicaciones del sistema operativo que puedan estar comprometidas.

Uno de los principales retos que ha tenido este trabajo ha sido la investigación y documentación dentro de un terreno desconocido: el kernel de Linux. Gracias al estudio realizado podemos dar respuesta a las siguientes preguntas: ¿dónde se encuentran los ficheros que guardan la información de un proceso? ¿Cómo puedo saber qué conexiones de red tiene activas el equipo? ¿Qué módulos tiene cargado el sistema? ¿Cómo Linux es capaz de ligar una conexión de red con un proceso? ¿Qué ficheros albergan información sobre conexiones SSH aceptadas? En general, hay que saber interpretar toda esa cantidad de información que se encuentra en el kernel y almacenarla en estructuras de datos para poder presentarla al usuario de forma organizada.

Otra barrera que se ha afrontado ha sido la tecnológica, en concreto con el lenguaje de programación Java. Muchas veces se sabe lo que se tiene que hacer, qué algoritmo seguir, pero no es nada trivial conseguir una implementación que sea aceptable en términos de eficiencia y eficacia. La implementación de la interfaz gráfica con la que interactúa el usuario, realizada a través del framework Swing de Java, tiene sus limitaciones, al igual que cualquier solución en el mundo de la programación. Por consiguiente, otro dilema que surge es descubrir hasta dónde puedes ir, hasta qué punto la idea se puede materializar en un programa.

El apartado relacionado con la estimación de un presupuesto de realización del proyecto ha resultado interesante debido a la dificultad que entraña, pues es necesario conocer muy bien la tecnología con la que se va a trabajar, el equipo de personas que van a desempeñar ese trabajo, tiempo, costes, etc. Si fuera un equipo de desarrollo de unas 10

personas que tienen que cobrar a fin de mes y cumplir las exigencias acordadas con un cliente, la responsabilidad como jefe de proyecto sería máxima.

De cara al mundo laboral, que ya es una realidad en mi vida, me va a ayudar mucho con algo que sólo da el tiempo y el esfuerzo: la experiencia.

8.2. Líneas futuras

Durante la realización de este proyecto han surgido líneas de trabajo muy interesantes que podrían ser continuadas en el futuro:

- Mejoras de funcionalidad:
    - Sistema de monitorización continua. Instalar un servidor web en las máquinas para monitorizar de forma remota usando los propios ficheros XML que ya genera la aplicación. Se podrían guardar en disco de forma periódica y mediante un script generar un histórico accesible vía navegador. La página web podría generarse con un script PHP o transformar el XML en HTML mediante XSLT. La monitorización puede llevar asociado un sistema de alertas que avise mediante correo electrónico o SMS de eventos sospechosos, por ejemplo una conexión SSH aceptada desde una IP de Rumanía. Mediante este sistema se podría montar una plataforma para monitorizar varias máquinas de forma remota como *Zabbix* o *M/Monit*.
    - En el listado de conexiones SSH sería posible añadir una nueva columna que indique el país de la dirección IP. La resolución IP – País se podría hacer mediante una consulta Whois. Durante la fase de implementación del proyecto se intentó abordar este apartado pero por problemas generados por la migración de los servidores de Whois a IPv6 no pudo resolverse en el tiempo estipulado.
    - Exportar ficheros en otros formatos distintos a XML. Formatos tabulados como CSV[40] o YML[41], o ficheros XLS.
    - Integración con la herramienta Volatility. Volatility es un proyecto que extrae una imagen de la memoria de un sistema en ejecución a fichero. Sobre ese fichero, con Volatility se pueden realizar consultas para ver

[40] *Comma-separated values*. Tipo de documento de formato abierto sencillo para representar datos en forma de tabla. Por ejemplo “987,juan,87345,10 norte 342”. Los caracteres separadores pueden variar.
[41] *YAML* es un formato de serialización de datos legible por el ojo humano.

procesos en ejecución, sockets de red abiertos, conexiones de red abiertas, librerías cargadas por cada proceso, etc. Existen dos ramas de desarrollo, una para sistemas *Windows* y otra para *Linux*.

    - Opción de cargar los ficheros generados en XML con la herramienta y poder visualizarlos de forma estática.
    - Pestaña con información de los ficheros abiertos (comando lsof).

- Mejoras de interfaz:
    - El usuario puede elegir las columnas que quiere ver en cada tabla. Pulsando con el botón derecho sobre la cabecera de la columna podría aparecer un listado de las columnas disponibles con un check delante.
- Mejoras de compatibilidad:
    - Adaptación a otros sistemas operativos gracias a la propiedad de multiplataforma de Java. Habría que modificar la parte de acceso al núcleo para la recuperación de información para cada máquina. Los sistemas UNIX parten con ventaja ya que Linux se basa en él.
- Optimización:
    - Cambios en la implementación que supongan mejoras de rendimiento de memoria.
    - Reducción del tamaño de la aplicación para facilitar su portabilidad mediante técnicas de compresión.
    - Creación de un fichero de configuración en el que parametrizar variables de la aplicación, como el periodo de refresco de datos para no saturar máquinas lentas.
- Difusión:
    - Difusión de la herramienta a través de distribuciones Linux especializadas en auditoría de seguridad como *BackTrack*.
    - Foros de Seguridad de RedIRIS.

9. Anexos

I. Representación de procesos en el kernel

En las páginas del manual de *www.kernel.org* se puede consultar mucha información sobre la representación de procesos en un kernel de Linux.

En las páginas del manual el sistema de ficheros proc [7] se define como un pseudo-sistema de ficheros el cual es usado como una interfaz para las estructuras de datos del kernel. Normalmente se monta en /proc. La gran mayoría de los ficheros que cuelgan del directorio /proc son de solo-lectura.

Dentro de /proc los procesos se organizan por directorios según su PID[42] de la forma: /proc/[PID]. En el fichero /proc/[PID]/status podemos encontrar mucha información sobre el estado de un proceso. Un ejemplo del contenido de ese fichero se muestra a continuación:

[42] PID: process identificator, es un número entero positivo que identifica a un proceso de forma unívoca en el kernel.

```
$ cat /proc/3515/status

Name:   bash
State:  S (sleeping)
Tgid:   3515
Pid:    3515
PPid:   3452
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    100     100     100     100
FDSize: 256
Groups: 16 33 100
VmPeak:     9136 kB
VmSize:     7896 kB
VmLck:         0 kB
VmHWM:      7572 kB
VmRSS:      6316 kB
VmData:     5224 kB
VmStk:        88 kB
VmExe:       572 kB
VmLib:      1708 kB
VmPTE:        20 kB
Threads:        1
SigQ:   0/3067
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000010000
SigIgn: 0000000000384004
SigCgt: 000000004b813efb
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed:   00000001
Cpus_allowed_list:      0
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        150
nonvoluntary ctxt switches:     545
```

Ilustración 39: Ejemplo /proc/3515/status

Los campos son los siguientes:

- Name: comando arrancado por este proceso
- State: estado actual del proceso. Toma los valores "R (running)", "S (sleeping)", "D (disk sleep)", "T (stopped)", "T (tracing stop)", "Z (zombie)", o "X (dead)".
- Tgid: identificador del grupo del thread. Si es un proceso con un solo thread coincidirá con el PID.
- Pid: identificador del proceso.
- PPid: PID del padre.

- TracerPid: identificador del proceso que está trazando este proceso. En caso de no existir será 0.
- Uid: cuatro dígitos correspondientes al identificador de usuario real[43], efectivo[44], saved set y sistema de ficheros.
- Gid: cuatro dígitos correspondientes al identificador de grupo real, efectivo, saved set y sistema de ficheros.
- FDSize: número de slots de descriptores de fichero actualmente asignados.
- Groups: lista de grupo suplementaria.
- VmPeak: tamaño máximo de memoria virtual.
- VmSize: tamaño de memoria virtual.
- VmLck: tamaño de memoria bloqueada.
- VmHWM: tamaño residente máximo establecido.
- VmRSS: tamaño residente establecido.
- VmData, VmStk, VmExe: tamaño de datos, pila y segmentos de texto, respectivamente.
- VmLib: tamaño de librerías compartidas.
- VmPTE: tabla de entradas de tamaño de página.
- Threads: número de threads in proceso que contienen este thread.
- SigPnd, ShdPnd: número de señales pendientes para el thread y para el proceso en su conjunto.
- SigBlk, SigIgn, SigCgt: máscaras indicando señales que están siendo bloqueadas, ignoradas y capturadas.
- CapInh, CapPrm, CapEff: máscaras de capacidades habilitadas en conjuntos heredables, permitidos y eficaces.
- CapBnd: capacidad de límite establecido.
- Cpus_allowed: máscara de procesadores en los cuales este proceso podría ejecutarse.
- Cpus_allowed_list: igual que el campo anterior pero en formato lista.
- Mems_allowed: máscara de nodos de memoria asignados para este proceso.
- Mems_allowed_list: igual que el campo anterior pero en formato lista.

[43] El usuario real corresponde al usuario que ha invocado al proceso.

[44] El usuario efectivo indica los privilegios con los que se está ejecutando un proceso. Puede ser distinto al usuario real. Por ejemplo el comando para cambiar la contraseña de usuario passwd tiene siempre como usuario efectivo root porque necesita modificar el fichero /etc/shadow, pero su usuario real es el que invocó el comando.

- Voluntary_context_switches, nonvoluntary_context_switches: número de cambios de context voluntaries e involuntarios.

II. Representación de las conexiones de red en el kernel

En Linux, la información relativa a las conexiones de red cuelga de /proc/net, en concreto en /proc/net/tcp y /proc/net/udp se pueden ver los sockets de red abiertos para los protocolos TCP y UDP respectivamente. A continuación se pueden observar unos fragmentos de ejemplo de cada fichero:

```
$ cat /proc/net/tcp

  sl  local_address rem_address   st tx_queue rx_queue tr tm-
>when retrnsmt   uid  timeout inode
   0: 00000000:0050 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 7750 1 f35b4a00 300 0 0 2 -1
   1: 00000000:0016 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 6782 1 f35b4000 300 0 0 2 -1
   2: 0100007F:0277 00000000:0000 0A 00000000:00000000
00:00000000 00000000     0        0 2937355 1 e1678000 300 0 0 2 -1
   3: 00000000:445C 00000000:0000 0A 00000000:00000000
00:00000000 00000000  1000        0 299996 1 f2d26800 300 0 0 2 -1
   4: 0100007F:0CEA 00000000:0000 0A 00000000:00000000
00:00000000 00000000   116        0 8400 1 f35b4f00 300 0 0 2 -1
   5: D18375A3:DC26 72C381AE:01BB 08 00000000:00000026
00:00000000 00000000  1000        0 1187067 1 e167a300 36 4 28 3 2
```

Ilustración 40: Ejemplo /proc/net/tcp

```
$ cat /proc/net/udp

  sl  local_address rem_address   st tx_queue rx_queue tr tm-
>when retrnsmt   uid  timeout inode ref pointer drops
   30: 00000000:89AD 00000000:0000 07 00000000:00000000
00:00000000 00000000   104        0 6912 2 f3d88280 0
  205: 00000000:445C 00000000:0000 07 00000000:00000000
00:00000000 00000000  1000        0 299992 2 f3d89180 0
  346: 00000000:14E9 00000000:0000 07 00000000:00000000
00:00000000 00000000   104        0 6910 2 f3d88000 0
```

Ilustración 41: Ejemplo /proc/net/udp

El formato de ambos ficheros es similar. Cada fila de estos ficheros muestra una conexión de red, a excepción de la primera que especifica el contenido de cada columna. Los campos son los siguientes:

- sl: número de la línea en el listado de salida.

- local_address: dirección IP local y número de puerto para el socket. La dirección IP se muestra en 4 bytes hexadecimales en formato little-endian[45]. El número de puerto es un simple par de bytes en hexadecimal. A continuación se muestra el algoritmo seguido para traducir la dirección IP:
    1. Cogemos el dato de entrada: C801A8C0:BDF2.
    2. Como la dirección está en formato little-endian hay que darle la vuelta: C0A801C8:BDF2.
    3. Separamos por puntos: C0.A8.01.C8:BDF2.
    4. Se traducen los valores hexadecimales: 192.168.1.200:48626.
- rem_address: dirección IP remota y el número de puerto para el socket. La codificación es la misma de antes.
- st: estado del socket. La transición de estados en un socket TCP es la siguiente:

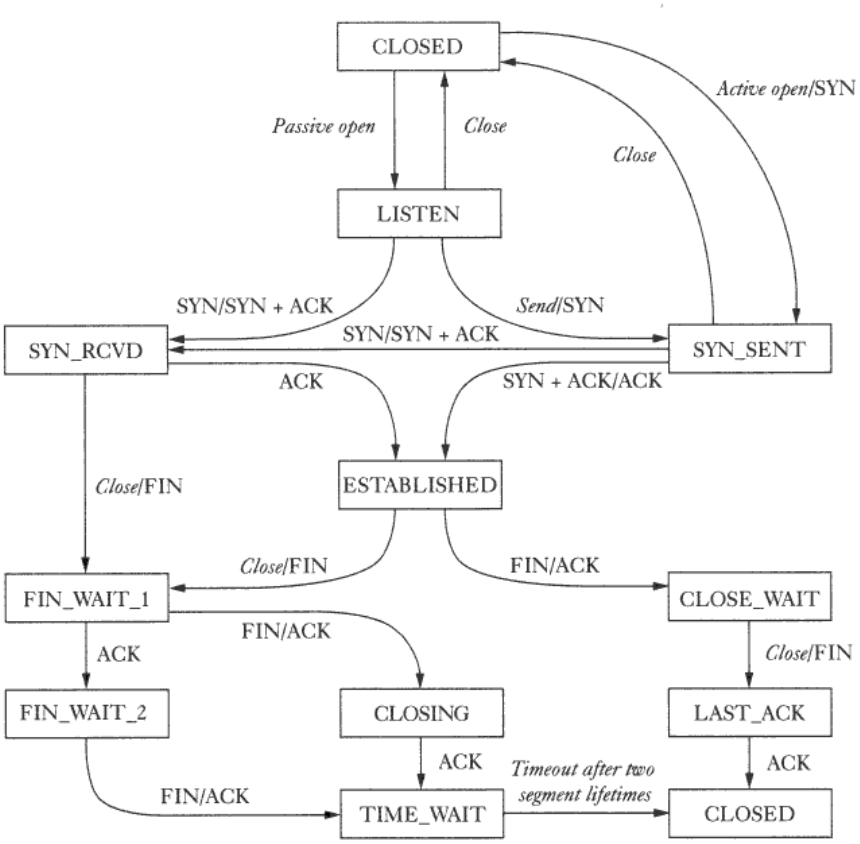


Ilustración 42: Diagrama de transición de estados TCP

[45] Little-endian es un formato de codificación en el que el byte menos significativo es el primero por lo que es necesario invertir el orden de los bytes para facilitar su compresión.

Este campo toma un valor hexadecimal definido en el siguiente enumerado[46]:

```
STATUS      -     HEX VALUE
ESTABLISHED -     01
SYN_SENT    -     02
SYN_RECV    -     03
FIN_WAIT1   -     04
FIN_WAIT2   -     05
TIME_WAIT   -     06
CLOSE       -     07
CLOSE_WAIT  -     08
LAST_ACK    -     09
```

Ilustración 43: Correspondencia estado - valor hexadecimal de un socket

UDP es un protocolo no orientado a conexión por lo que el diagrama se reduce a ESTABLISHED y CLOSE.

- tx_queue: tamaño de la cola de transmisión.
- rx_queue: tamaño de la cola de recepción.
- tr: indica si un temporizador está activado para ese socket. El valor cero indica que no está activado.
- tm->when: indica el tiempo restante para el timeout del temporizador.
- retrnsmt: sin uso.
- uid: ID del usuario propietario del socket. Es el mismo ID que se encuentra en el fichero /etc/passwd.
- timeout: sin uso.
- inode: inodo, número que identifica el socket en el sistema de ficheros.

[46] Definido en la interfaz include/net/tcp_states.h del kernel de Linux desde la versión 2.6. En versiones anteriores como 2.4 se puede encontrar en include/Linux/tcp.h. Consultado en TOMOYO Linux: Cross reference of source code

III. Representación de módulos cargados en el kernel

Los módulos [8] cargados en el kernel (en inglés Linux kernel modules o LKM), en un sistema Linux, son los controladores que, a parte del kernel, realizan la interacción final con los dispositivos. Dicho de otra forma, un dispositivo sólo podrá ser usado si el kernel lo soporta o existe un módulo capaz de controlarlo[47].

En un sistema Linux podemos encontrar un listado de todos los módulos cargados en el kernel en el fichero /proc/modules. Un ejemplo del contenido de este fichero se muestra a continuación:

```
$ cat /proc/modules

nfs      170109  0 -          Live 0x129b0000
lockd    51593   1 nfs,       Live 0x128b0000
nls_utf8 1729    0 -          Live 0x12830000
vfat     12097   0 -          Live 0x12823000
fat      38881   1 vfat,      Live 0x1287b000
autofs4  20293   2 -          Live 0x1284f000
sunrpc   140453  3 nfs,lockd, Live 0x12954000
3c59x    33257   0 -          Live 0x12871000
uhci_hcd 28377   0 -          Live 0x12869000
md5      3777    1 -          Live 0x1282c000
ipv6     211845 16 -          Live 0x128de000
ext3     92585   2 -          Live 0x12886000
jbd      65625   1 ext3,      Live 0x12857000
dm mod   46677   3 -          Live 0x12833000
```

Ilustración 44: Ejemplo /proc/modules

- La primera columna contiene el nombre del módulo.
- La segunda columna indica el tamaño del módulo en memoria. Se expresa en bytes.
- La tercera columna lista cuantas instancias del módulo están actualmente cargadas. El valor 0 representa un módulo descargado[48].
- La cuarta columna muestra si el módulo depende de otros módulos para ser usado.
- La quinta columna muestra el estado del módulo. Los valores posibles son: Live, Loading o Unloading.

[47] En Windows, el concepto de módulo sería análogo el concepto de "driver" o controlador de un dispositivo.

[48] Descargado, del inglés unloaded, se refiere a que en el actual arranque del kernel había sido usado al menos por un proceso.

- La sexta columna muestra el actual offset[49] de memoria en el kernel para el módulo cargado. Esa información puede ser usada a menudo para depuración.

[49] El término offset, en informática, significa desplazamiento. Si el espacio de memoria del kernel comienza en la dirección X, el módulo estará en X+OFFSET.

IV. Conexiones SSH en Linux

SSH (Secure Shell, en español: intérprete de órdenes segura) es un protocolo que sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante intérprete de comandos. Desde SSH es posible copiar datos de forma segura, gestionar claves RSA y pasar los datos de cualquier aplicación por un canal seguro tunelizado.

Este protocolo trabaja de forma similar a Telnet. De cara a la seguridad, la principal diferencia es que SSH usa técnicas de cifrado que hacen que la información que viaja por la red vaya de manera no legible ocultando usuario, contraseña y comandos que se escriben en la sesión. Esto hace imposible el uso de un sniffer[50] de red para analizar la comunicación.

En Linux cuando se acepta una conexión SSH se escribe una línea de log en /var/log/auth.log[51]. Además, en este fichero se registran logins del sistema, las veces que se hace su[52], intentos fallidos, etc.

```
$ tail /var/log/auth.log

Aug 10 10:39:01 sissomah CRON[6973]: pam_unix(cron:session):
session opened for user root by (uid=0)
Aug 10 10:39:01 sissomah CRON[6973]: pam_unix(cron:session):
session closed for user root
Aug 10 11:09:01 sissomah CRON[7126]: pam_unix(cron:session):
session opened for user root by (uid=0)
Aug 10 11:09:01 sissomah CRON[7126]: pam_unix(cron:session):
session closed for user root
Aug 10 11:17:01 sissomah CRON[7135]: pam_unix(cron:session):
session opened for user root by (uid=0)
Aug 10 11:17:01 sissomah CRON[7135]: pam_unix(cron:session):
session closed for user root
Aug 10 11:20:10 sissomah sshd[7141]: Accepted password for
jmlotero from ::1 port 42890 ssh2
```

Ilustración 45: Ejemplo /var/log/auth.log

[50] Un sniffer es un tipo de software que registra la información que envían los periféricos. Un sniffer de red analiza el tráfico de paquetes que pasar por su segmento de red, aún sin ir dirigidos a él (modo promiscuo).

[51] En Linux Debian/Ubuntu. En otras distribuciones puede cambiar. Por ejemplo, Red Hat los almacena en /var/log/secure.

[52] El comando su se usa para cambiar de usuario, incluido root. Es necesario saber la contraseña. Cambia el ID efectivo del usuario y del grupo.

V. XML generado por la aplicación

A continuación se muestra, a modo ejemplo, el contenido de un fichero XML generado por la aplicación para el apartado de conexiones de red:

```
<?xml version="1.0" encoding="UTF-8"?>
<modules>
	<module name="Net">
		<row num="0">
			<column name="proto">tcp</column>
			<column name="sl">0</column>
			<column name="local_ip">0.0.0.0</column>
			<column name="local_port">22</column>
			<column name="rem_ip">0.0.0.0</column>
			<column name="rem_port">*</column>
			<column name="st">LISTEN</column>
			<column name="tx_queue">00000000</column>
			<column name="rx_queue">00000000</column>
			<column name="tr">00</column>
			<column name="tm->when">00000000</column>
			<column name="uid">0</column>
			<column name="inode">6868</column>
		</row>
		<row num="1">
			<column name="proto">tcp</column>
			<column name="sl">1</column>
			<column name="local_ip">127.0.0.1</column>
			<column name="local_port">631</column>
			<column name="rem_ip">0.0.0.0</column>
			<column name="rem_port">*</column>
			<column name="st">LISTEN</column>
			<column name="tx_queue">00000000</column>
			<column name="rx_queue">00000000</column>
			<column name="tr">00</column>
			<column name="tm->when">00000000</column>
			<column name="uid">0</column>
			<column name="inode">7192</column>
		</row>
		<row num="2">
			<column name="proto">tcp</column>
			<column name="sl">2</column>
			<column name="local_ip">0.0.0.0</column>
			<column name="local_port">17500</column>
			<column name="rem_ip">0.0.0.0</column>
			<column name="rem_port">*</column>
			<column name="st">LISTEN</column>
			<column name="tx_queue">00000000</column>
			<column name="rx_queue">00000000</column>
			<column name="tr">00</column>
			<column name="tm->when">00000000</column>
			<column name="uid">1000</column>
			<column name="inode">719112</column>
		</row>
		<row num="3">
			<column name="proto">tcp</column>
			<column name="sl">3</column>
			<column name="local_ip">127.0.0.1</column>
			<column name="local_port">3306</column>
			<column name="rem_ip">0.0.0.0</column>
			<column name="rem_port">*</column>
```

```
                        <column name="st">LISTEN</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">116</column>
                        <column name="inode">8445</column>
                  </row>
                  <row num="4">
                        <column name="proto">tcp</column>
                        <column name="sl">4</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">50529</column>
                        <column name="rem_ip">163.117.136.131</column>
                        <column name="rem_port">22</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">02</column>
                        <column name="tm->when">0007BBDC</column>
                        <column name="uid">1000</column>
                        <column name="inode">115536</column>
                  </row>
                  <row num="5">
                        <column name="proto">tcp</column>
                        <column name="sl">5</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">47562</column>
                        <column name="rem_ip">199.47.216.173</column>
                        <column name="rem_port">443</column>
                        <column name="st">CLOSE_WAIT</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000026</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">3475864</column>
                  </row>
                  <row num="6">
                        <column name="proto">tcp</column>
                        <column name="sl">6</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">40417</column>
                        <column name="rem_ip">163.117.131.195</column>
                        <column name="rem_port">17500</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">721054</column>
                  </row>
                  <row num="7">
                        <column name="proto">tcp</column>
                        <column name="sl">7</column>
                        <column
name="local_ip">163.117.131.209</column>
```

```
                        <column name="local_port">50305</column>
                        <column name="rem_ip">163.117.176.175</column>
                        <column name="rem_port">22</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">02</column>
                        <column name="tm->when">0008A246</column>
                        <column name="uid">1000</column>
                        <column name="inode">252763</column>
                  </row>
                  <row num="8">
                        <column name="proto">tcp</column>
                        <column name="sl">8</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">48435</column>
                        <column name="rem_ip">163.117.136.181</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">4891491</column>
                  </row>
                  <row num="9">
                        <column name="proto">tcp</column>
                        <column name="sl">9</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">44299</column>
                        <column name="rem_ip">163.117.136.180</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">22614</column>
                  </row>
                  <row num="10">
                        <column name="proto">tcp</column>
                        <column name="sl">10</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">44276</column>
                        <column name="rem_ip">163.117.136.180</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">18597</column>
                  </row>
                  <row num="11">
                        <column name="proto">tcp</column>
```

```
                        <column name="sl">11</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">34357</column>
                        <column name="rem_ip">163.117.136.181</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">36011</column>
                  </row>
                  <row num="12">
                        <column name="proto">tcp</column>
                        <column name="sl">12</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">34358</column>
                        <column name="rem_ip">163.117.136.181</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">36041</column>
                  </row>
                  <row num="13">
                        <column name="proto">tcp</column>
                        <column name="sl">13</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">44292</column>
                        <column name="rem_ip">163.117.136.180</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">21507</column>
                  </row>
                  <row num="14">
                        <column name="proto">tcp</column>
                        <column name="sl">14</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">53092</column>
                        <column name="rem_ip">74.125.230.184</column>
                        <column name="rem_port">443</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">02</column>
                        <column name="tm->when">000000BD</column>
                        <column name="uid">1000</column>
                        <column name="inode">4255249</column>
```

```
                </row>
                <row num="15">
                        <column name="proto">tcp</column>
                        <column name="sl">15</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">36126</column>
                        <column name="rem_ip">75.126.110.108</column>
                        <column name="rem_port">443</column>
                        <column name="st">CLOSE_WAIT</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000026</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">3055800</column>
                </row>
                <row num="16">
                        <column name="proto">tcp</column>
                        <column name="sl">16</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">56525</column>
                        <column name="rem_ip">163.117.136.132</column>
                        <column name="rem_port">22</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">02</column>
                        <column name="tm->when">0005089E</column>
                        <column name="uid">1000</column>
                        <column name="inode">111987</column>
                </row>
                <row num="17">
                        <column name="proto">tcp</column>
                        <column name="sl">17</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">36479</column>
                        <column name="rem_ip">50.16.230.238</column>
                        <column name="rem_port">443</column>
                        <column name="st">CLOSE_WAIT</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000026</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">719334</column>
                </row>
                <row num="18">
                        <column name="proto">tcp</column>
                        <column name="sl">18</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">34359</column>
                        <column name="rem_ip">163.117.136.181</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
```

```
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">36047</column>
                  </row>
                  <row num="19">
                        <column name="proto">tcp</column>
                        <column name="sl">19</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">44284</column>
                        <column name="rem_ip">163.117.136.180</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">19864</column>
                  </row>
                  <row num="20">
                        <column name="proto">tcp</column>
                        <column name="sl">20</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">35329</column>
                        <column name="rem_ip">199.47.217.148</column>
                        <column name="rem_port">80</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">719242</column>
                  </row>
                  <row num="21">
                        <column name="proto">tcp</column>
                        <column name="sl">21</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">57723</column>
                        <column name="rem_ip">163.117.131.195</column>
                        <column name="rem_port">3389</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">450093</column>
                  </row>
                  <row num="22">
                        <column name="proto">tcp</column>
                        <column name="sl">22</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">44283</column>
                        <column name="rem_ip">163.117.136.180</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
```

```
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">19831</column>
                  </row>
                  <row num="23">
                        <column name="proto">tcp</column>
                        <column name="sl">23</column>
                        <column
name="local_ip">163.117.131.209</column>
                        <column name="local_port">34326</column>
                        <column name="rem_ip">163.117.136.181</column>
                        <column name="rem_port">993</column>
                        <column name="st">ESTABLISHED</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">18593</column>
                  </row>
                  <row num="24">
                        <column name="proto">udp</column>
                        <column name="sl">205</column>
                        <column name="local_ip">0.0.0.0</column>
                        <column name="local_port">17500</column>
                        <column name="rem_ip">0.0.0.0</column>
                        <column name="rem_port">*</column>
                        <column name="st">CLOSE</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">1000</column>
                        <column name="inode">719108</column>
                  </row>
                  <row num="25">
                        <column name="proto">udp</column>
                        <column name="sl">346</column>
                        <column name="local_ip">0.0.0.0</column>
                        <column name="local_port">5353</column>
                        <column name="rem_ip">0.0.0.0</column>
                        <column name="rem_port">*</column>
                        <column name="st">CLOSE</column>
                        <column name="tx_queue">00000000</column>
                        <column name="rx_queue">00000000</column>
                        <column name="tr">00</column>
                        <column name="tm->when">00000000</column>
                        <column name="uid">104</column>
                        <column name="inode">7026</column>
                  </row>
                  <row num="26">
                        <column name="proto">udp</column>
                        <column name="sl">484</column>
                        <column name="local_ip">0.0.0.0</column>
                        <column name="local_port">54131</column>
                        <column name="rem_ip">0.0.0.0</column>
                        <column name="rem_port">*</column>
                        <column name="st">CLOSE</column>
```

```
                    <column name="tx_queue">00000000</column>
                    <column name="rx_queue">00000000</column>
                    <column name="tr">00</column>
                    <column name="tm->when">00000000</column>
                    <column name="uid">104</column>
                    <column name="inode">7028</column>
                </row>
        </module>
</modules>
```

VI. Script run.sh

El siguiente script se encarga de lanzar la herramienta con privilegios de superusuario usando el JRE incluido en la aplicación.

```
$ cat run.sh

#!/bin/bash
sudo ./jre/bin/java -jar "dist/Warroning2.jar"
```

Ilustración 46: Script run.sh

VII. Manual de usuario

En este apartado se describe el manual de usuario que tiene por objetivo enseñar al usuario cómo hacer uso de la aplicación de la forma más sencilla posible.

a. Funciones generales

De forma general, la estructura de la interfaz se desglosa en una parte de opciones generales situada en el margen superior de la ventana y un panel de pestañas que presenta la información. Cada pestaña contiene un módulo. Se han desarrollado los correspondientes a procesos, conexiones de red, módulos cargados en el kernel y conexiones SSH aceptadas.

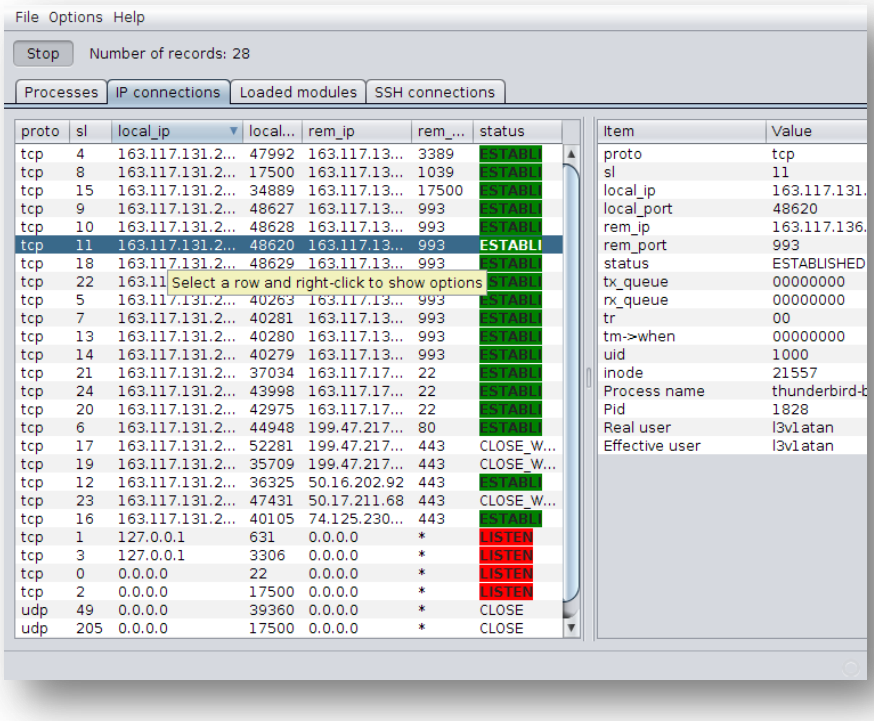


Ilustración 47: MU - Vista principal

De forma más detallada, la aplicación está compuesta por los siguientes componentes:

- **Menú principal:** se encuentra en la parte superior de la ventana. Está agrupado en tres desplegables: desde File podemos salir de la aplicación con la opción Exit; en Options podemos generar un XML global con los datos contenidos en todas los módulos; en Help se presenta información variada en el apartado About.

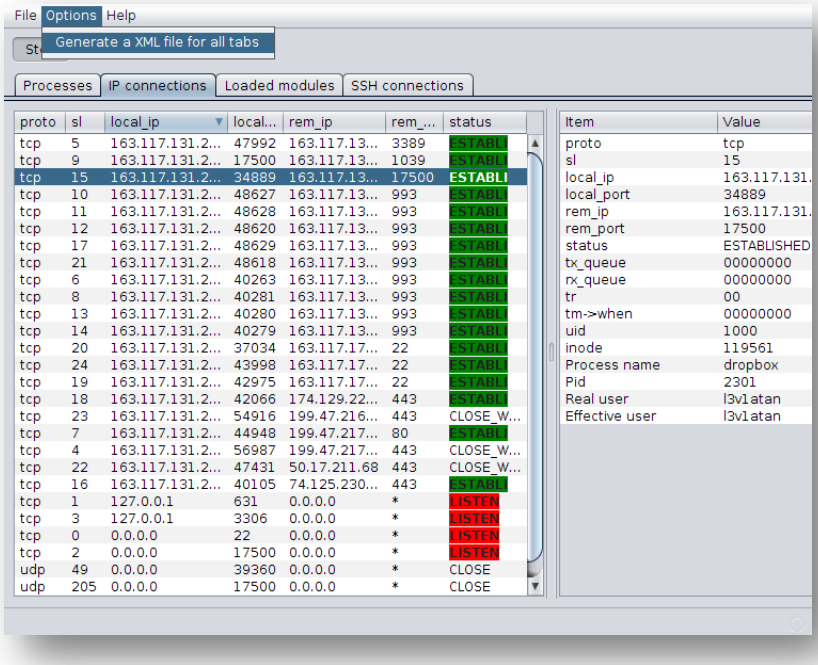


Ilustración 48: MU Menú Options

- **Botón start/stop:** habilita o deshabilita el autorefresco de los datos mostrados en las pestañas.
- **Etiqueta "Number of records":** indica el número de entradas que hay en la tabla que se está mostrando.
- **Panel de pestañas:** cada sección tiene asignado un módulo de la aplicación y se explicarán con detalle en los siguientes puntos. Para cada uno se puede generar un fichero XML pinchando con el botón derecho sobre el listado y seleccionando la opción "Save a XML file for this tab".

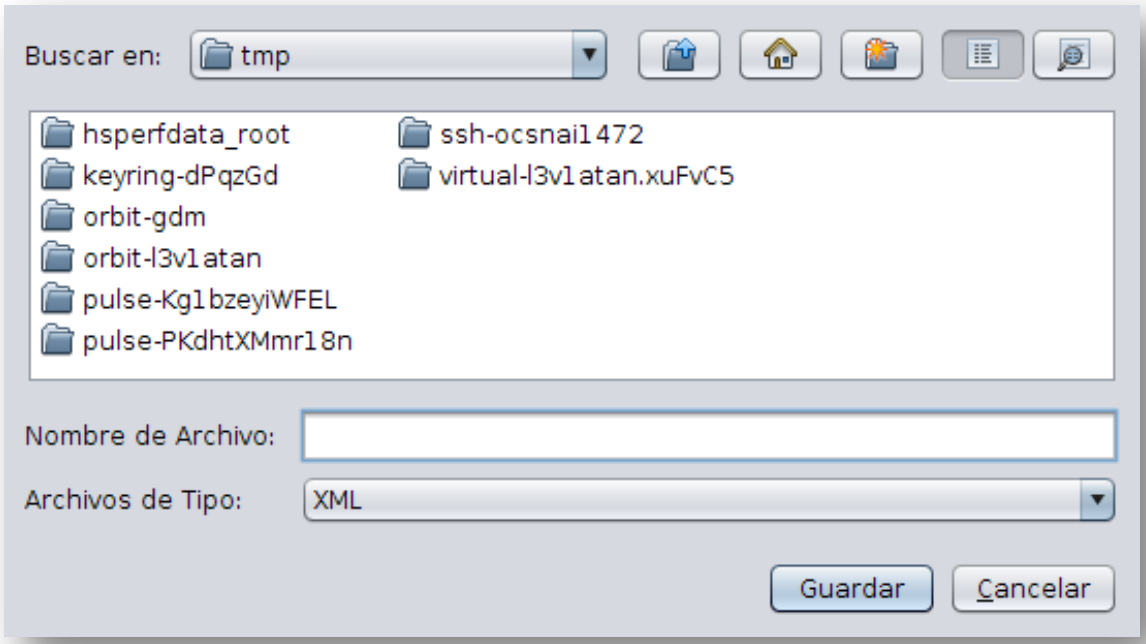


Ilustración 49: MU Guardar fichero XML

b. Procesos

Desde esta sección obtiene toda la información sobre los procesos que se encuentran en memoria.

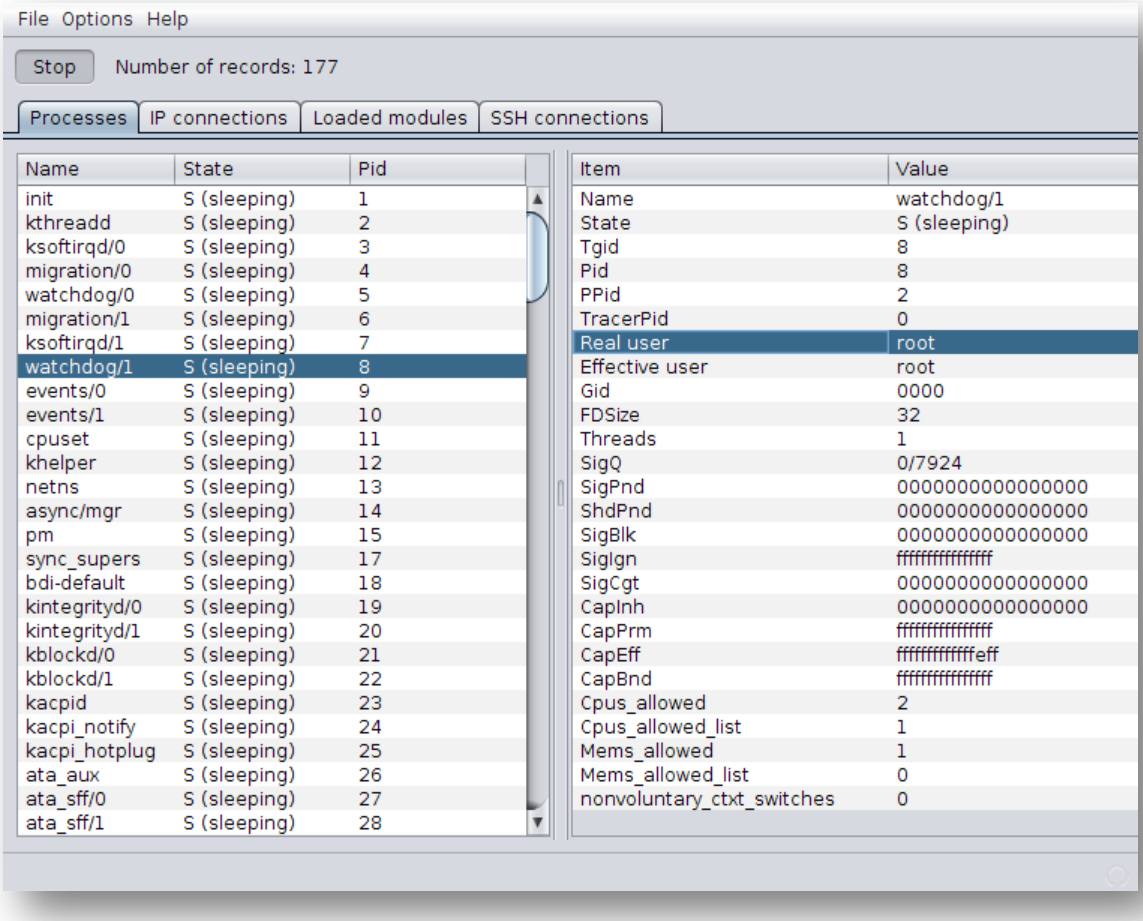


Ilustración 50: MU Vista de procesos

Esta vista consta de dos tablas. En la izquierda podemos ver un listado de procesos en los que se especifica su nombre, su estado y su identificador de proceso (columnas Name, State y PID[53], respectivamente). En la derecha se encuentra la información detallada del proceso seleccionado del listado anterior mediante el par elemento-valor (columnas Item y Value).

[53] PID: process identificator, es un número entero positivo que identifica a un proceso de forma unívoca en el kernel.

c. Conexiones de red

En esta pestaña se muestra la información sobre las conexiones de red que se encuentran activas.

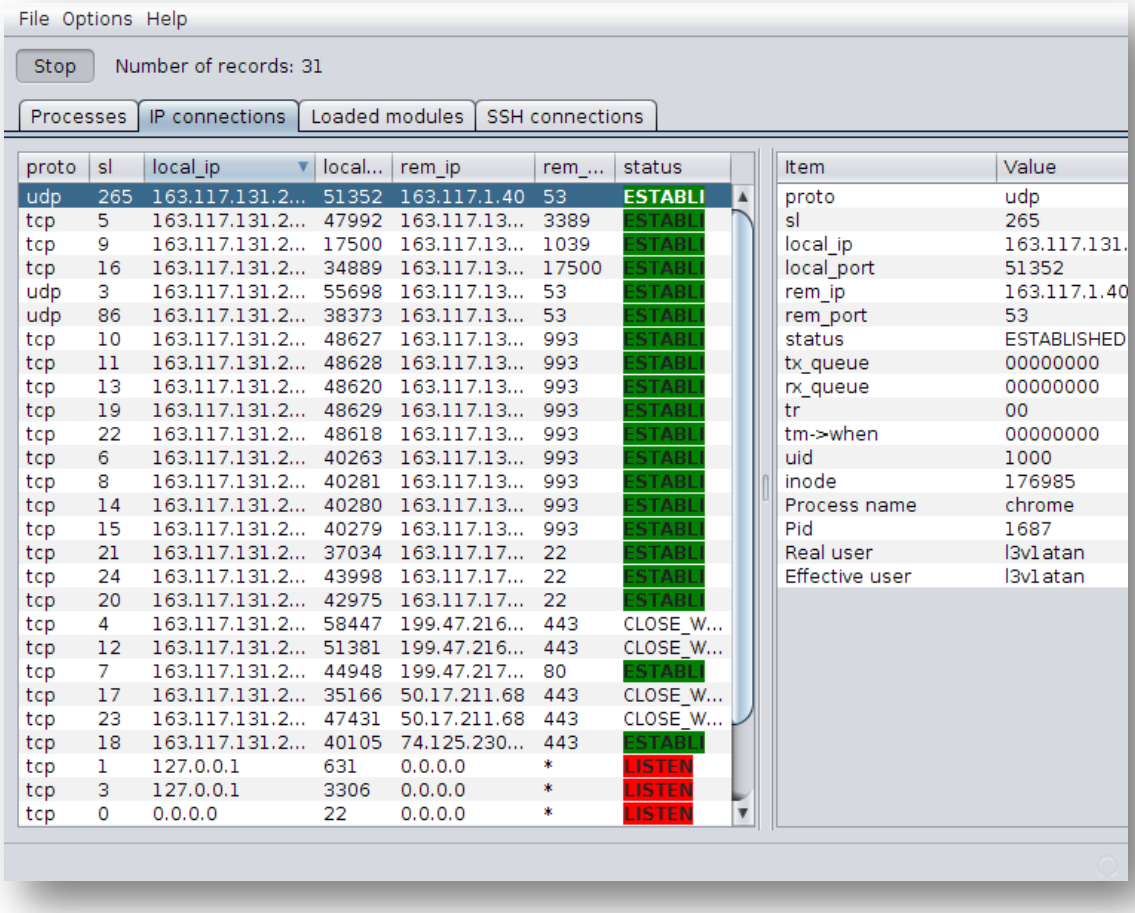


Ilustración 51: MU Vista de conexiones de red

Esta vista consta de dos tablas. En la izquierda podemos ver un listado de las conexiones de red con información general, entre las que se pueden destacar las direcciones IP y puertos involucrados, estado de la conexión, protocolo usado, etc. En la derecha se encuentra la información detallada del proceso seleccionado del listado anterior mediante el par elemento-valor (columnas Item y Value).

Existe la posibilidad de realizar una navegación de la conexión de la pestaña de conexiones de red al proceso de la pestaña de procesos. Si accedemos al menú emergente después de seleccionar una entrada de la tabla se muestra la opción “Navigate to process” con la que se puede realizar la acción.

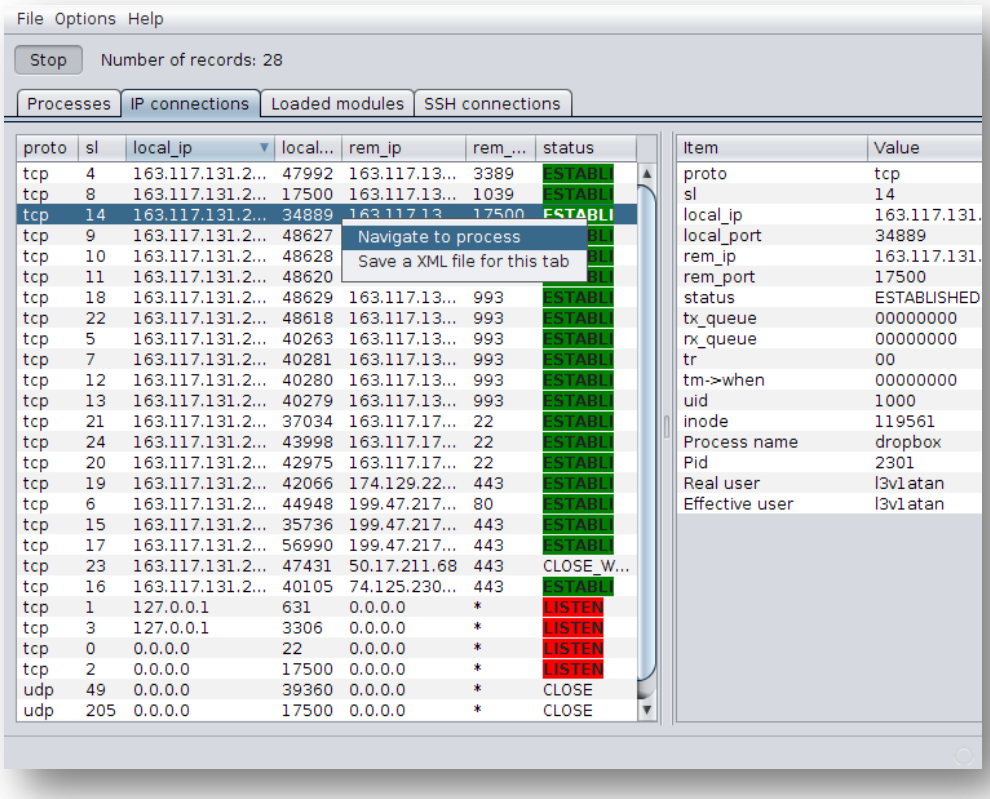
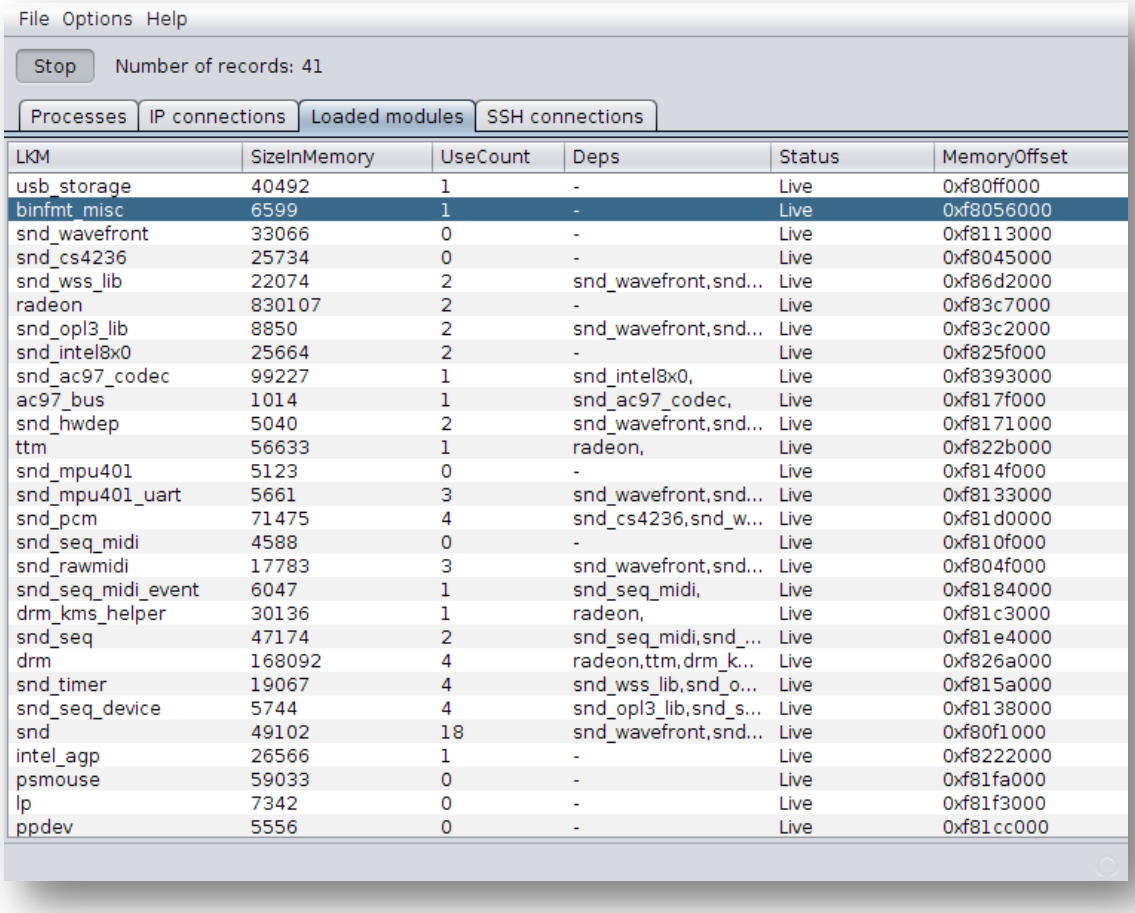


Ilustración 52: MU Menú emergente

d. Módulos cargados

Desde este apartado se obtiene toda la información relativa a los módulos cargados en el kernel (Linux Kernel Modules, LKM).



LKM	SizeInMemory	UseCount	Deps	Status	MemoryOffset
usb_storage	40492	1	-	Live	0xf80ff000
binfmt_misc	6599	1	-	Live	0xf8056000
snd_wavefront	33066	0	-	Live	0xf8113000
snd_cs4236	25734	0	-	Live	0xf8045000
snd_wss_lib	22074	2	snd_wavefront,snd...	Live	0xf86d2000
radeon	830107	2	-	Live	0xf83c7000
snd_opl3_lib	8850	2	snd_wavefront,snd...	Live	0xf83c2000
snd_intel8x0	25664	2	-	Live	0xf825f000
snd_ac97_codec	99227	1	snd_intel8x0,	Live	0xf8393000
ac97_bus	1014	1	snd_ac97_codec,	Live	0xf817f000
snd_hwdep	5040	2	snd_wavefront,snd...	Live	0xf8171000
ttm	56633	1	radeon,	Live	0xf822b000
snd_mpu401	5123	0	-	Live	0xf814f000
snd_mpu401_uart	5661	3	snd_wavefront,snd...	Live	0xf8133000
snd_pcm	71475	4	snd_cs4236,snd_w...	Live	0xf81d0000
snd_seq_midi	4588	0	-	Live	0xf810f000
snd_rawmidi	17783	3	snd_wavefront,snd...	Live	0xf804f000
snd_seq_midi_event	6047	1	snd_seq_midi,	Live	0xf8184000
drm_kms_helper	30136	1	radeon,	Live	0xf81c3000
snd_seq	47174	2	snd_seq_midi,snd_...	Live	0xf81e4000
drm	168092	4	radeon,ttm,drm_k...	Live	0xf826a000
snd_timer	19067	4	snd_wss_lib,snd_o...	Live	0xf815a000
snd_seq_device	5744	4	snd_opl3_lib,snd_s...	Live	0xf8138000
snd	49102	18	snd_wavefront,snd...	Live	0xf80f1000
intel_agp	26566	1	-	Live	0xf8222000
psmouse	59033	0	-	Live	0xf81fa000
lp	7342	0	-	Live	0xf81f3000
ppdev	5556	0	-	Live	0xf81cc000

Ilustración 53: MU Vista de módulos cargados

Los datos se muestran en una única tabla en la que se puede ver un listado de módulos en el que para cada uno se especifica: nombre del módulo, tamaño que ocupan en memoria, número de instancias del módulo cargadas, módulos de los que depende, estado y offset (desplazamiento) de memoria (columnas LKM, SizeInMemory, UseCount, Deps, Status y MemoryOffset, respectivamente).

e. Conexiones establecidas vía SSH

Este módulo se encarga de mostrar la información relativa a las conexiones aceptadas vía SSH.

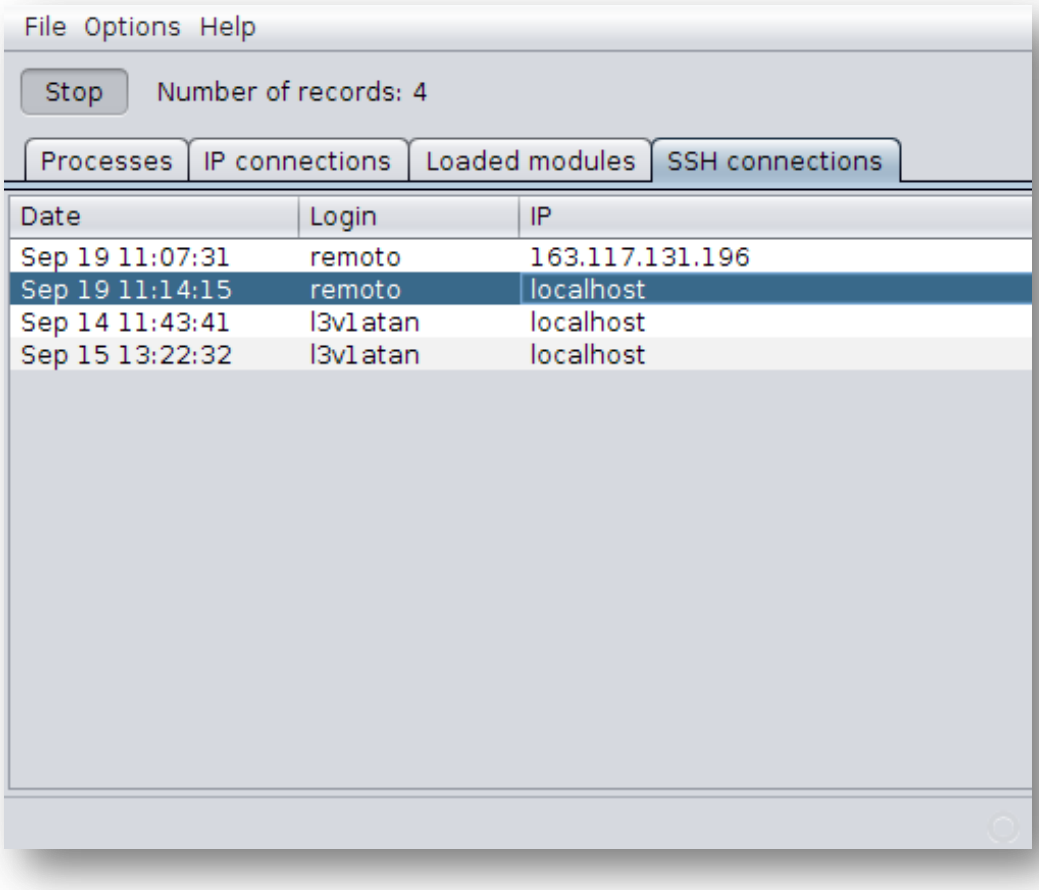


Ilustración 54: MU Vista de conexiones establecidas vía SSH

Por cada fila se encuentras los siguientes datos:

- **Date:** fecha en la que se realizó la conexión.
- **Login:** login con el que se accedió.
- **IP:** dirección IP desde la que se conectó.

Si se pulsa con el botón derecho sobre una entrada aparecerá un menú emergente que mostrará los elementos “Save a XML file for this tab” (explicado anteriormente) y “Show .bash_history”.

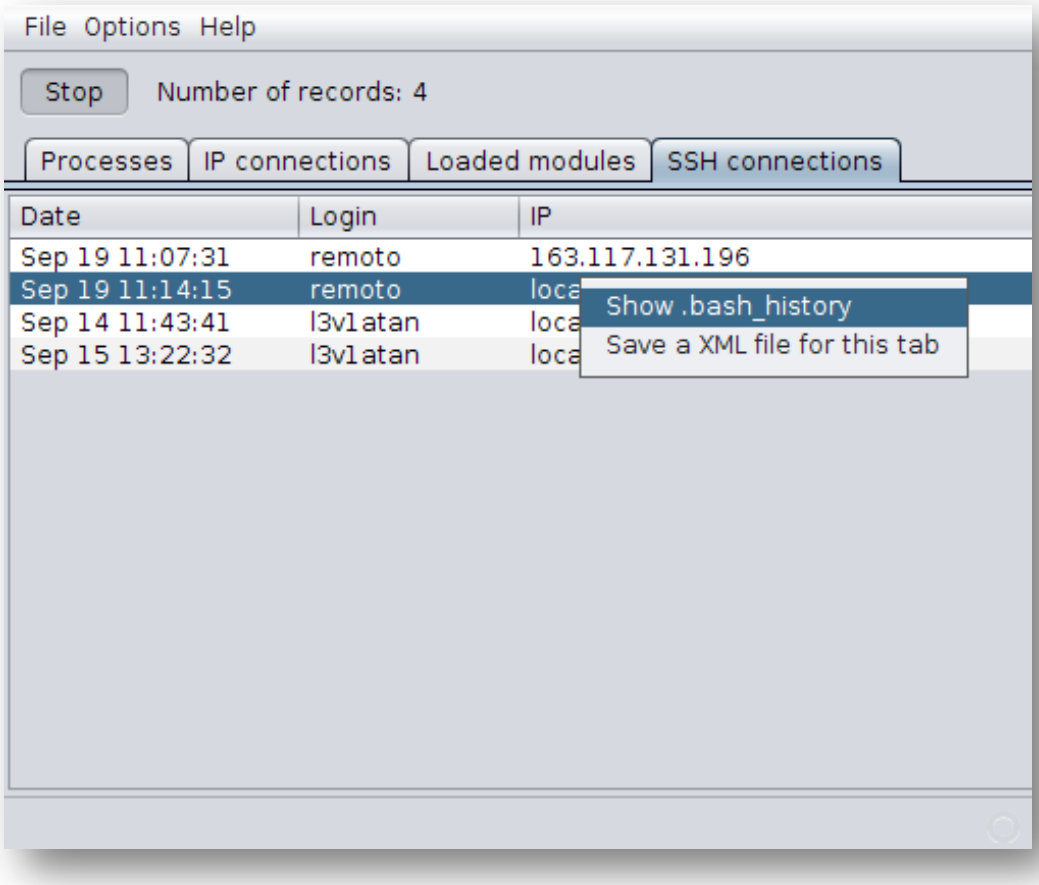


Ilustración 55: MU Menú emergente

Si se selecciona la opción "Show .bash_history" aparecerá una ventana nueva en la que se mostrará el fichero de log de comandos .bash_history para ese usuario. En la ilustración anterior se puede apreciar cómo se selecciona una entrada del usuario "remoto". En la siguiente se puede contemplar el histórico de comandos de "remoto".

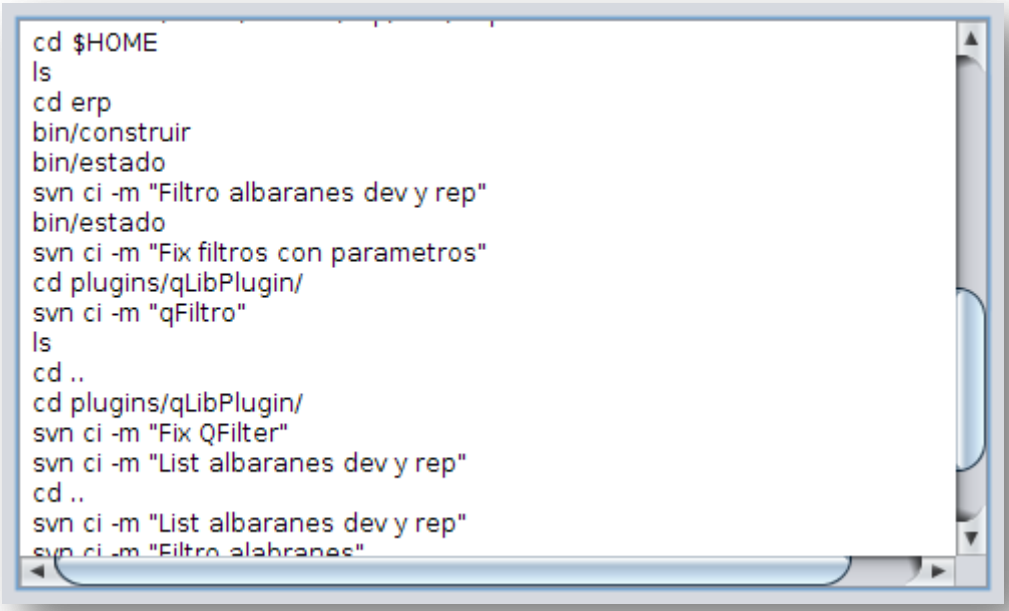


Ilustración 56: MU Vista del .bash_history

El .bash_history es un fichero que recopila un histórico de los comandos que el usuario ha introducido en el terminal.

10.Bibliografía

Para la realización del proyecto se han consultado los siguientes libros:

- *"Understanding the Linux kernel (3rd edition)"*, Daniel P. Bovet. Publicado por O'Reilly Media. Noviembre 2005. ISBN 0596005652.
- *"Linux kernel development (2nd edition)"*, Robert Love. Publicado por Novell. Enero 2005. ISBN 0672327201.
- *"The Linux kernel premier: a top-down approach for x86 and PowerPC architectures"*, Claudia Salzberg Rodriguez, Gordon Fischer, Steven Smolski. Publicado por Prentice Hall. Septiembre 2005. ISBN 0131181637.
- *"Fedora 10 and Red Hat Enterprise Linux Bible"*, Christopher Negus. Publicado por Wiley. Enero 2009. ISBN 0470413395.
- *"Ubuntu Unleashed 2011 Edition: Covering 10.10 and 11.04 (6th edition)"*, Matthew Helmke, Andrew Hudson, Paul Hudson. Publicado por Sams. Diciembre 2010. ISBN 0672333449.

11.Referencias

A lo largo del documento se han ido citando una serie de referencias a fuentes de información externas al documento, procedentes de Internet. El formato que siguen dichas referencias es "[X]", siendo X el número de la referencia.

En la siguiente tabla se muestran todas las referencias citadas en el documento indicando el número de la referencia, el contenido, la fuente correspondiente (en este caso es una URL) y la última fecha de acceso a dicha referencia.

Referencia	Contenido	Fuente	Última fecha de acceso
[1]	Informe de incidencias de seguridad del año 2010 de RedIRIS	http://www.rediris.es/cert/doc/informes/2010/	lun, 16/05/11
[2]	Sysinternals	http://technet.microsoft.com/es-es/sysinternals/	jue, 30/06/11
[3]	What's Running	http://www.whatsrunning.net/	vie, 1/07/11
[4]	Plantilla presupuesto fin de carrera	https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_2.xlsx	jue, 12/05/11
[5]	Metodología ESA	http://www.arcos.inf.uc3m.es/~ii_si/estandarESA.zip	jue, 28/07/11
[6]	Metodología Métrica V3	http://www.csi.map.es/csi/metrica3	mié, 18/05/11
[7]	Proc(5) – Linux manual page	http://www.kernel.org/doc/man-pages/online/pages/man5/proc.5.html	sáb, 2/07/11
[8]	Kernel y módulos (definición, objetivos, herramientas)	http://structio.sourceforge.net/guias/AA_Linux_colegio/kernel-y-modulos.html	sáb, 2/07/11

Tabla 113: Referencias utilizadas